Chapter 4

Queueing Network Model Inputs and Outputs

4.1. Introduction

We are prepared now to state precisely the inputs and outputs of queueing network models. We noted in Chapter 1 that, in order to achieve an appropriate balance between accuracy and cost, we are restricting our attention to the subset of general networks of queues that consists of the *separable* queueing networks, extended where necessary for the accurate representation of particular computer system characteristics. Sections 4.2 - 4.4 describe the inputs and outputs of separable queueing networks. For notational simplicity we first present this material in the context of models with a single customer class (Sections 4.2 and 4.3), and then generalize to multiple class models (Section 4.4). In Section 4.5, we discuss certain computer system characteristics that cannot be represented directly using the inputs available for separable models, and certain performance measures that cannot be obtained directly from the available outputs. These motivate the extensions of separable networks that will be explored later in the book.

4.2. Model Inputs

The basic entities in queueing network models are *service centers*, which represent system resources, and *customers*, which represent users or jobs or transactions. Table 4.1 lists the inputs of single class queueing network models, which describe the relationships among customers and service centers. In the subsections that follow, these parameters are discussed in some detail.

4.2.1. Customer Description

The workload intensity may be described in any of three ways, named to suggest the computer system workloads they are best suited to representing:

customer description	The *workload intensity*, one of: λ, the *arrival rate* (for *transaction* workloads), or N, the *population* (for *batch* workloads), or N and Z, the *think time* (for *terminal* workloads)
center description	K, the number of *service centers* For each service center k: its *type*, either *queueing* or *delay*
service demands	For each service center k: $D_k \equiv V_k S_k$, the *service demand*

Table 4.1 – Single Class Model Inputs

- A *transaction* workload has its intensity specified by a parameter λ, indicating the rate at which requests (customers) arrive. A transaction workload has a population that varies over time. Customers that have completed service leave the model.
- A *batch* workload has its intensity specified by a parameter N, indicating the average number of active jobs (customers). (N need not be an integer.) A batch workload has a fixed population. Customers that have completed service can be thought of as leaving the model and being replaced instantaneously from a backlog of waiting jobs.
- A *terminal* workload has its intensity specified by two parameters: N, indicating the number of active terminals (customers), and Z, indicating the average length of time that customers use terminals ("think") between interactions. (Again, N need not be an integer.)

A terminal workload is similar to a batch workload in that its total population is fixed. In fact, a terminal workload with a think time of zero is in every way equivalent to a batch workload. On the other hand, a terminal workload is similar to a transaction workload in that the population of the *central subsystem* (the system excluding the terminals) varies, provided that the terminal workload has a non-zero think time. Note that N is an upper bound on the central subsystem population of a terminal workload, whereas no upper bound exists for a transaction workload.

We sometimes refer to models with transaction workloads as *open* models, since there is an infinite stream of arriving customers. Models with batch or terminal workloads are referred to as *closed* models, since customers "re-circulate". This distinction is made because the algorithms used to evaluate open models differ from those used for closed models. It highlights the similarity between batch and terminal workloads.

4.2.2. Center Description

Service centers may be of two types: *queueing* and *delay.* These are represented as shown in Figure 4.1.

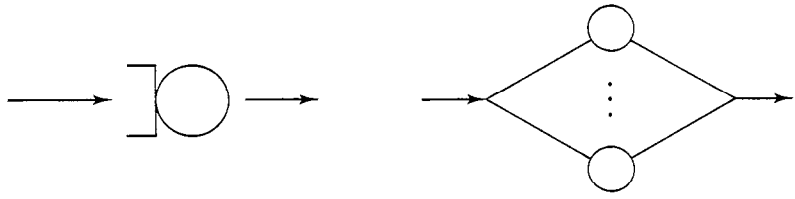

Figure 4.1 – Queueing and Delay Service Centers

Customers at a queueing center compete for the use of the server. Thus the time spent by a customer at a queueing center has two components: time spent waiting, and time spent receiving service. Queueing centers are used to represent any system resource at which users compete for service, e.g., the CPU and I/O devices. As shown in the figure, a queueing center is drawn as a queue plus a server.

Because customers in a single class model are indistinguishable, it is not necessary to specify the scheduling discipline at a queueing center. The same performance measures will result from any scheduling discipline in which exactly one customer is in service whenever there are customers at the center.

Customers at a delay center each (logically) are allocated their own server, so there is no competition for service. Thus the residence time of a customer at a delay center is exactly that customer's service demand there. The most common use of a delay center is to represent the think time of terminal workloads. However, delay centers are useful in any situation in which it is necessary to impose some known average delay. For instance, a delay center could be used to represent the delay incurred by sending large amounts of data over a dedicated low speed transmission line. As shown in the figure, an icon suggestive of concurrent activity is used to represent a delay center.

4.2.3. Service Demands

The service demand of a customer at center k, D_k, is the total amount of time the customer requires in service at that center. Thus the set of service demands (one for each center) characterizes the behavior of the customer in terms of processing requirements. In a single class model,

customers are indistinguishable with respect to their service demands, which can be thought of as representing the "average customer" in the actual system.

D_k can be calculated directly as B_k/C (the measured busy time of device k divided by the measured number of system completions), or may be thought of as the product of V_k, the number of visits that a customer makes to center k, and S_k, the service requirement per visit. It is possible to parameterize queueing network models at this more detailed level. However, a surprising characteristic of separable queueing networks is that their solutions depend only on the product of V_k and S_k at each center, and not on the individual values. Thus a model in which customers make 100 visits to the CPU, each for 10 milliseconds of service, is equivalent to one in which customers make a single visit for one second of service. For simplicity (to reduce the number of parameters and to facilitate obtaining their values) we generally will choose to parameterize our models in terms of D_k. Note that we define D to be the total service demand of a customer at all centers: $D \equiv \sum_{k=1}^{K} D_k$.

4.3. Model Outputs

Table 4.2 lists the outputs obtained by evaluating a single class queueing network model. Comments appear in the subsections that follow.

system measures	R average system response time X system throughput Q average number in system
center measures	U_k utilization of center k R_k average residence time at center k X_k throughput of center k Q_k average queue length at center k

Table 4.2 – Single Class Model Outputs

The values of these outputs depend upon the values of all of the model inputs. It will be especially useful to be able to specify that an output value corresponds to a particular workload intensity value. To do so, we follow the output with the parenthesized workload intensity: $X(N)$ is the throughput for a batch or terminal class with population N, $Q_k(\lambda)$ is the average queue length at center k for a transaction class with arrival rate λ, etc.

4.3.1. Utilization

The utilization of a center may be interpreted as the proportion of time the device is busy, or, equivalently, as the average number of customers in service there. (The latter interpretation is the only one that makes sense for a delay center.)

4.3.2. Residence Time

Just as D_k is the total service demand of a customer at center k (in contrast to S_k, the service requirement per visit), R_k is the total residence time of a customer at center k (as opposed to the time spent there on a single visit). If the model is parameterized in terms of V_k and S_k, then the time spent per visit at center k can be calculated as R_k/V_k.

System response time, R, corresponds to our intuitive notion of response time; for example, the interval between submitting a request and receiving a response on an interactive system. Obviously, system response time is the sum of the residence times at the various centers: $R = \sum_{k=1}^{K} R_k$.

4.3.3. Throughput

If a model is parameterized in terms of D_k then we can obtain system throughput, X, but do not have sufficient information to calculate device throughputs, X_k. (This is a small price to pay for the convenience that results from the less detailed parameterization.) If a model is parameterized in terms of V_k and S_k, then device throughputs can be calculated using the forced flow law, as $X_k = V_k X$.

4.3.4. Queue Length

The average queue length at center k, Q_k, includes all customers at that center, whether waiting or receiving service. The number of customers waiting can be calculated as $Q_k - U_k$, since U_k can be interpreted as the average number of customers receiving service at center k.

Q denotes the average number in system. For a batch class, $Q = N$. For a transaction class, $Q = XR$ (by Little's law). For a terminal class, $Q = N - XZ$ ($Q = XR$, and $R = N/X - Z$.) In general, the average population of any subsystem can be obtained either by multiplying the throughput of the subsystem by the residence time there, or by summing the queue lengths at the centers belonging to the subsystem.

4.3.5. Other Outputs

Various other outputs can be computed at some additional cost. As one example, we occasionally will wish to know the *queue length distribution* at a center: the proportion of time that the queue length has each possible value. We denote the proportion of time that the queue length at center k has the value i by $P[Q_k = i]$.

4.4. Multiple Class Models

4.4.1. Inputs

Multiple class models consist of C *customer classes*, each of which has its own workload intensity (λ_c, N_c, or N_c and Z_c) and its own service demand at each center ($D_{c,k}$). Within each class, the customers are indistinguishable. (Note that we have re-used the symbol C, which denoted customer completions in Chapter 3. Confusion will not arise.)

Multiple class models consisting entirely of open (transaction) classes are referred to as open models. Models consisting entirely of closed (batch or terminal) classes are referred to as closed. Models consisting of both types of classes are referred to as *mixed*.

The overall workload intensity of a multiple class model is described by a vector with an entry for each class: $\vec{\lambda} \equiv (\lambda_1, \lambda_2, \ldots, \lambda_C)$ if the model is open, $\vec{N} \equiv (N_1, N_2, \ldots, N_C)$ if it is closed (in point of fact, Z_c also must be included for terminal classes), and $\vec{I} \equiv (N_1 \text{ or } \lambda_1, N_2 \text{ or } \lambda_2, \ldots, N_C \text{ or } \lambda_C)$ if it is mixed.

As was the case for single class models, we do not specify the scheduling discipline at a queueing center. Roughly, the assumption made is that the scheduling discipline is *class independent*, i.e., it does not make use of information about the class to which a customer belongs. The same performance measures will result from any scheduling discipline that satisfies this assumption, along with the earlier assumption that exactly one customer is in service whenever there are customers at the center.

Table 4.3 summarizes the inputs of multiple class models. By analogy to the single class case, we define D_c to be the total service demand of a class c customer at all centers: $D_c \equiv \sum_{k=1}^{K} D_{c,k}$.

4.4.2. Outputs

All performance measures can be obtained on a per-class basis (e.g., $U_{c,k}$ and X_c) as well as on an aggregate basis (e.g., U_k and X). For utilization, queue length, and throughput, the aggregate performance measure

customer description	C, the number of *customer classes* For each class c: its *workload intensity*, one of: λ_c, the *arrival rate* (for *transaction* workloads), or N_c, the *population* (for *batch* workloads), or N_c and Z_c, the *think time* (for *terminal* workloads)
center description	K, the number of *service centers* For each service center k: its *type*, either *queueing* or *delay*
service demands	For each class c and center k: $D_{c,k} \equiv V_{c,k} S_{c,k}$, the *service demand*

Table 4.3 – Multiple Class Model Inputs

equals the sum of the per-class performance measures (e.g., $U_k = \sum_{c=1}^{C} U_{c,k}$). For residence time and system response time, however, the per-class measures must be weighted by relative throughput, as follows:

$$R = \sum_{c=1}^{C} \frac{R_c X_c}{X} \qquad R_k = \sum_{c=1}^{C} \frac{R_{c,k} X_c}{X}$$

This makes intuitive sense, and can be demonstrated formally using Little's law (see Exercise 2).

Table 4.4 summarizes the outputs of multiple class models. The following reminders, similar to comments made in the context of single class models, should be noted in studying the table:

- The basic outputs are average values (e.g., average response time) rather than distributional information (e.g., the 90th percentile of response time). Thus the word "average" should be understood even if it is omitted.
- X_k and $X_{c,k}$ are meaningful only if the model is parameterized in terms of $V_{c,k}$ and $S_{c,k}$, rather than $D_{c,k}$.
- To specify that an output value corresponds to a particular workload intensity value, we follow the output symbol with the parenthesized workload intensity.

system measures	**aggregate**	R average system response time X system throughput Q average number in system
	per class	R_c average class c system response time X_c class c system throughput Q_c average class c number in system
center measures	**aggregate**	U_k utilization of center k R_k average residence time at center k X_k throughput at center k Q_k average queue length at center k
	per class	$U_{c,k}$ class c utilization of center k $R_{c,k}$ average class c residence time at center k $X_{c,k}$ class c throughput at center k $Q_{c,k}$ average class c queue length at center k

Table 4.4 – Multiple Class Model Outputs

4.5. Discussion

The specific inputs and outputs available for separable queueing network models, as just described, are dictated by a set of mathematical assumptions imposed to ensure efficiency of evaluation. The purpose of the present section is to consider the practical impact of these assumptions on the accuracy of our models. Specifically:

- What important computer system characteristics cannot be represented directly using separable models?
- Given these apparent inadequacies, how can we explain the success of separable models in computer system analysis?
- How does the analyst approach the inevitable situations in which separable models truly are inadequate?

Naturally, complete answers to these questions must await the remainder of the book. The present section contains a foreshadowing of these answers, to provide insight and guide intuition. For simplicity, our discussion will be set largely in the single class context.

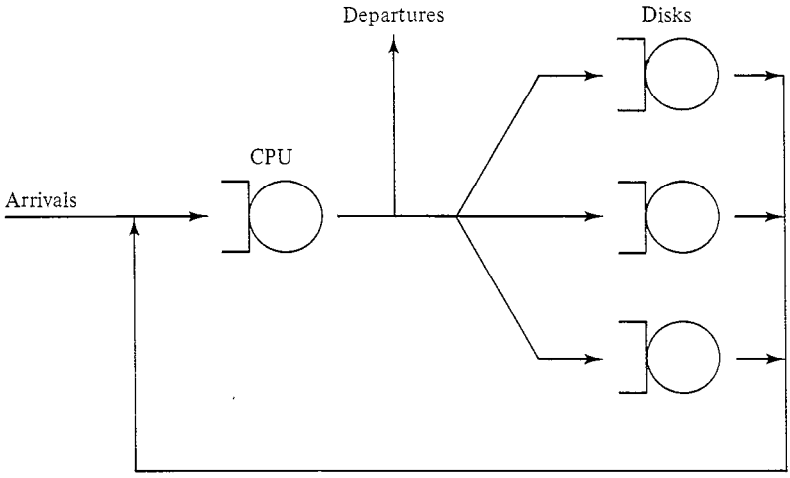

Figure 4.2 – The Canonical Computer System Model

Figure 4.2 illustrates the canonical separable queueing network model of a centralized system, which appears throughout the book. This model has the inputs and outputs discussed earlier in this chapter. Service centers are used to represent the CPU and the active I/O storage devices, e.g., disks. On the one hand, this model bears a close structural resemblance to a computer system. On the other hand, there are certain computer system characteristics that cannot be represented directly using the available inputs, and certain performance measures that cannot be obtained directly from the available outputs. These include:

- *simultaneous resource possession* – We have no direct way to express the fact that a customer may require simultaneous service at multiple resources. As an example, in order to transfer data to or from disk it may be necessary to concurrently use the disk, a controller, and a channel.
- *memory constraints* – Using a transaction workload, we are assuming implicitly that an arbitrarily large number of customers can be memory resident simultaneously. Using a batch workload, we are assuming implicitly that the multiprogramming level is constant. Using a terminal workload, we are assuming implicitly that all terminal users can be resident in memory simultaneously. In practice, it often occurs that the number of simultaneously active jobs varies over time but is limited by some memory constraint.

- *blocking* — In systems such as store-and-forward communications networks, the state of one resource can affect the processing of customers at another.
- *adaptive behavior* — A timesharing system may dynamically allocate scratch files to lightly loaded disks. A communications network may make dynamic routing decisions based on the populations at various nodes.
- *process creation and synchronization* — Since the number of customers in a class must either remain constant (closed classes) or be unbounded (open classes), it is not possible to represent explicitly a process executing a *fork* (spawning a sub-process) when it reaches a particular point in its computation. Similarly, since customers are independent of one another it is not possible to model directly synchronization points in the computation of two or more processes.
- *high service time variability* — In practice, extremely high service burst length variability can degrade the performance of a system.
- *priority scheduling* — Since a priority scheduler makes use of class dependent information, it will yield different performance measures than the class independent scheduling disciplines assumed in multiple class queueing network models.
- *response time distributions* — The list of useful model outputs obtainable directly at reasonable cost does not include the distribution of response times.

How is it, then, that separable queueing network models are successful at representing the behavior of complex contemporary computer systems, and at projecting the impact of modifications to their hardware, software, and workload?

First, consider the process of defining and parameterizing a model of an existing system. Much of the relevant complexity of the system that we appear to be ignoring is, in fact, captured *implicitly* in the measurement data used to parameterize the model. As an example, consider the effect of I/O path contention. Our canonical model represents only disks, not intermediate path elements such as channels and controllers. However, in parameterizing the model we will set the service demand at each disk center k, D_k, equal to the measured total disk busy time per job, which we will calculate as $U_k \; T/C$ (C here is the measured number of completions). In measuring the disk, we will find it busy not only during seek, latency, and data transfer, but also during those periods when it is attempting to obtain a free path. In other words, the effect of I/O path contention is incorporated *indirectly*, through the disk service demand parameters. A model parameterized in this way can be expected to do a good job of representing the behavior of the system during the measurement interval.

Next, consider using such a model to project the effect of modifications. In many cases, indirect representations of system characteristics based on measurement data can be assumed to be insensitive to the proposed modification. For example, the primary effect of a CPU upgrade can be represented in a model by adjusting CPU service demands. Any effect of this modification on disk service demands — either "intrinsic" demands (seek, latency, and data transfer times) or the component due to path contention — is strictly secondary in nature. It is in these cases that separable models prove adequate on their own.

Sometimes, of course, the objective of a study is to answer detailed questions about modifications that can be expected to affect the indirect representations of system characteristics. For example, if I/O path contention were known to be a significant problem, an analyst might want to use a queueing network model to project the performance improvement that would result from path modifications. In cases such as this, separable models can be augmented with procedures that calculate revised estimates for those portions of various service demands that are indirect representations of relevant system characteristics. These are the "extensions" alluded to in Chapter 1. This approach achieves the necessary accuracy, while preserving the ability to evaluate the model efficiently. Such techniques exist for each of the system characteristics mentioned earlier in this section.

4.6. Summary

We have enumerated and discussed the inputs and the outputs of separable queueing network models. These were summarized for the single class case in Tables 4.1 and 4.2, respectively, and for the multiple class case in Tables 4.3 and 4.4, respectively.

We have noted that the availability of inputs and outputs is dictated by assumptions imposed to ensure the efficient evaluation of the model. We have considered the practical impact of these assumptions on the accuracy of the models.

In many cases, separable models are adequate by themselves, because complex system characteristics are captured implicitly in the measurement data used to parameterize them. Part II of the book is devoted to evaluation algorithms for models of this sort.

In other cases, separable models must be augmented with procedures that calculate revised estimates for those portions of various service demands that are indirect representations of relevant system characteristics. Part III of the book is devoted to such procedures.

4.7. Exercises

1. Consider the system with which you are most familiar:
   a. How would you obtain parameter values for a single class model from the available measurement data?
   b. How would you obtain parameter values for a multiple class model from the available measurement data?
   c. What aspects of your system important to its performance seem to be omitted from the simple single or multiple class models that you might define?
2. Show that $R = \sum_{c=1}^{C} \frac{R_c X_c}{X}$, that is, that the average response time in a system with multiple job classes is a throughput-weighted average of the individual average response times.
3. In creating a model of a computer system, there are two extreme positions we can take as to the workload representation:
   a. assume all jobs are identical, in which case a single class model is appropriate, or
   b. assume each job is significantly different from every other job, and represent the workload with a class per job.

   What are the advantages and disadvantages of each approach?