QUEUEING NETWORKS[†]

Peter J. Denning, Naval Postgraduate School, Monterey, California

January 2008
Rev 7/7/08

Abstract: Queueing networks model workflow in distributed computing systems and networks. They predict throughput, queue length, response time, and bottlenecks with simple, fast algorithms.

Keywords: queueing networks, performance calculation, performance prediction, operational laws, throughput, response time, bottlenecks, mean value analysis, capacity planning

A major airline has set up a computerized transaction system used by its ticket agents to sell seats on its aircraft. The airline has authorized 1000 agents around the country to make reservations from their workstations. A "disk farm"--- a large collection of magnetic-disk storage devices --- in New York contains all the records of flights, routes, and reservations. On average, each agent issues a transaction against this database once every 60 seconds. One disk contains a directory that is consulted during every transaction to locate other disks that contain the actual data; on average, each transaction accesses the directory disk 10 times. The directory disk takes an average of five milliseconds to service each request, and it is busy 80% of the time.

On the basis of this information, can we calculate the throughput and response time of this system? Can we find the bottlenecks? Can we say what happens to the response time if we reduce directory disk visits by half or doubled the number of agents?

These performance questions are typical. The answers help analysts decide whether a system can perform well under the load offered, and where to add capacity if it is too slow. Most people think that no meaningful answers can be given without detailed knowledge of the system structure --- the locations and types of the agents' workstations, the communication bandwidth between each workstation and the disk farm, the number and types of disks in the farm, access patterns for the disks, local processors and random-access memory within the farm, the type of operating system, the types of transactions, and more. It may

---

[†] This article is adapted from two articles published by the author in *American Scientist* magazine in 1991: (a) Queueing in Networks of Computers, May-June, 206-209, and (b) In the Queue: Mean Values, September-October, 402-403.

come as a surprise, therefore, that the first two questions --- concerning throughput, response time, and bottlenecks --- can be answered precisely from the information given. The second two questions --- concerning effects of configuration changes --- can be answered with reasonable estimates made from the information given and a few plausible assumptions.

These questions illustrate the two important types of performance questions: calculation and prediction. Calculation questions seek metrics in the same observation period where parameters were measured. Prediction questions provide metrics in a different (future) observation period from when parameters were measured.

Operational Laws

A computer network is composed of interconnected servers. Servers include workstations, disks, processors, databases, printers, displays, and any other devices that can carry out computational tasks. Each server receives and queues up messages from other servers that specify tasks for the server to carry out; a typical message might ask a server to run a computationally intensive program, to perform an input-output transaction, or to access a database. A transaction is a specified sequence of tasks submitted to the network; when a server completes a particular task, it deletes the request from its queue and sends a request to another server to perform the next task in the same transaction.

Measurements of servers are always made during a definite observation period. Basic measures typically include event counters and timers. These and other measures derived from them are called operational quantities. Invariant relations among operations quantities that hold in every observation period are called operational laws.

By counting outgoing messages and by measuring the time that a server's queue is nonempty, it is easy to measure the output rate X, the mean service time S, and the utilization U of a server. These three empirical quantities satisfy the relation $U = SX$, known as the utilization law (Fig. 1). Similarly, by measuring the "space-time" accumulated by queued tasks, it is easy to determine the mean queue length Q and the mean response time R: These quantities satisfy the relation $Q = RX$, which is known as Little's Law (Fig. 2).

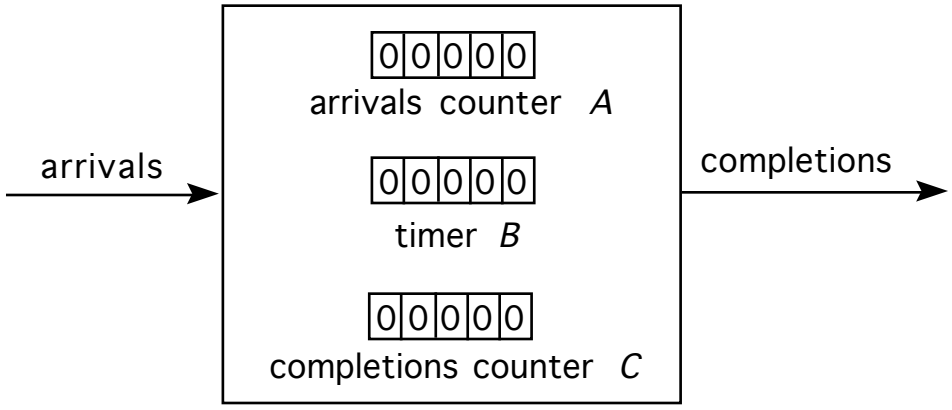

Figure 1. The task-processing server is the basic element of a network of computers. Over an observation period of length T, the counter A records the number of tasks that arrive at the server, the counter C records the number of tasks completed, and the timer B measures the total busy time (time when tasks are present). The utilization of the server is $U = B/T$, the output rate is $X = C/T$, and the mean service time per completed task is $S = B/C$. The identity $B/T = (C/T)(B/C)$ translates to the utilization law: $U = XS$. Because utilization cannot exceed 1, the output rate cannot exceed the saturation rate of $1/S$.

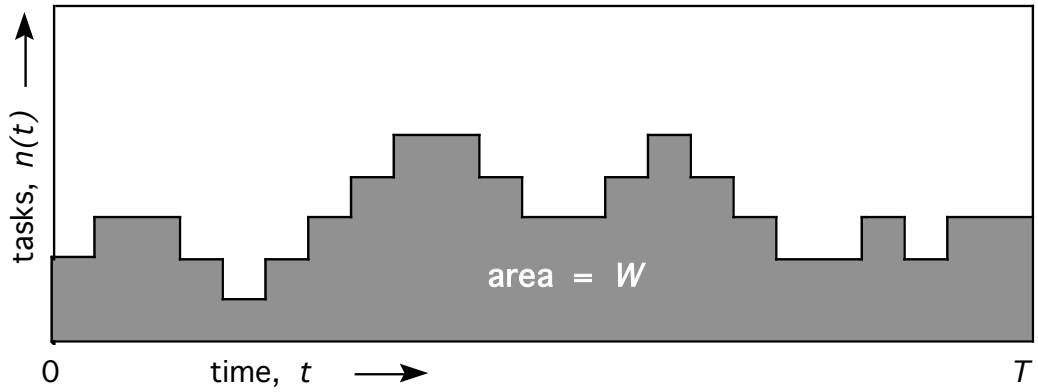

Figure 2. The average response time of a server can be calculated from just a few measurements. Let $n(t)$ denote the number of tasks in the server at time t. Let W denote the area under the graph of $n(t)$ in the interval from time 0 to time T; W is the number of task-seconds of accumulated waiting. The mean number of tasks at the server is $Q = W/T$, and the mean response time per completed task is $R = W/C$. The identity $W/T = (C/T)(W/C)$ translates to Little's law: $Q = XR$. The mean service time S and the mean response time R are not the same; R includes queueing delay as well as service time.

The utilization law and Little's law are counterparts of well-known limit theorems for stochastic queueing systems in a steady state. These theorems will usually be verified in actual measurements, not because a steady state has been attained, but because the measured quantities obey the operational laws (*1*,*2*).

The tasks that make up a transaction can be regarded as a sequence of visits by the transaction to the servers of the network. The average number of visits per transaction to a particular server i is called the visit ratio V_i for that server; the server's output rate X_i and the system's output rate X_0 satisfy the relation $X_i = V_i X_0$, which is known as the forced-flow law (Fig. 3). This remarkable law shows that knowledge of the visit ratios and the output rate of any one server is sufficient to determine the output rates of every other server and of the system itself. Moreover, any two networks with the same visit ratios have the same flows, no matter what is the interconnection structure among their servers.

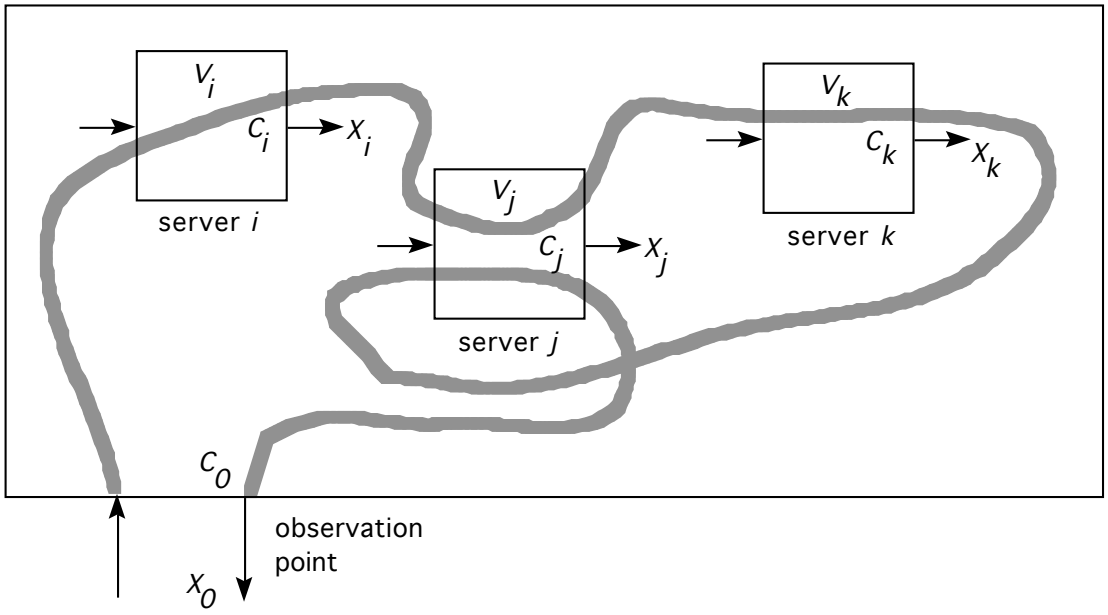

Figure 3. Flow of transactions through a network of servers can be calculated from a few selected measurements. Over an observation period T, the system completes C_0 transactions. The average number of tasks per transaction for server i is $V_i = C_i/C_0$; V_i is called the visit ratio because each task is regarded as a "visit" by the transaction to the server. Here the transaction visits servers i and k once and server j twice. The identity $C/T = (C_i/C_0)(C_0/T)$ translates to the forced-flow law: $X_i = V_i X_0$. This law says that the task flow at one point in the system determines the task flows everywhere. This law holds regardless of the interconnections among the servers; any two networks with the same visit ratios will have the same flows. The local throughput constraint $X_i \le 1/S_i$ translates to a system throughput constraint $X_0 \le 1/V_i S_i$.

In a network, a server's output is a portion of another server's input or of the system's output. It simplifies an analysis to assume that the input and output flows of a server are identical --- a condition known as flow balance. Strictly speaking, "throughput" refers to the rate of a flow-balanced server. The definitions do not imply flow balance. In most real systems, a bound is placed on the number of tasks that can be in the system at once; as long as the number of completions at every server is large compared with this bound, the error introduced by assuming flow balance will be negligible. For this reason, flow balance does not generally introduce much error into the models.

When a network of servers receives all of its requests from a finite population of N users who each delay an average of Z seconds until submitting a new transaction, the response time for a request in the network satisfies the response-time formula $R = N/X_0 - Z$ (Fig. 4). This formula is exact for flow balance.

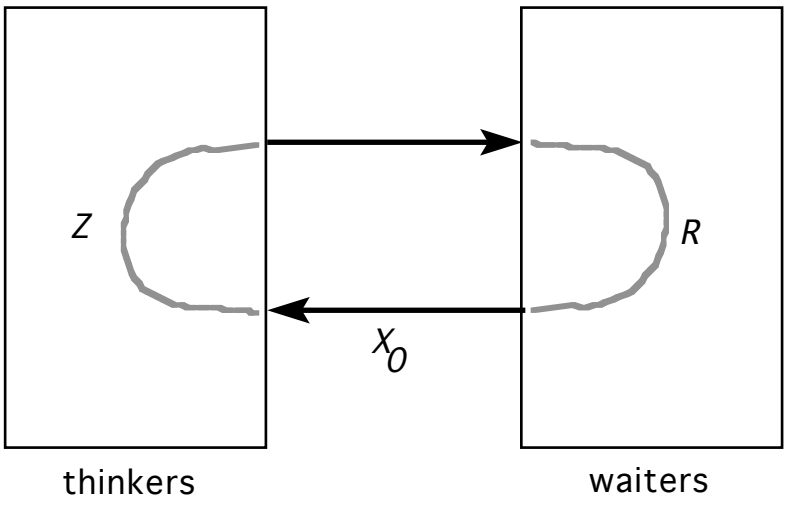

Figure 4. Users of a transaction system alternate between periods of "thinking" and "waiting" when using the system. The total number of thinkers and waiters is the number of users N. The average (waiting) response time per transaction is R and the average thinking time is Z. Little's law says that the mean number of active users in an entire system is equal to the mean cycle time of the system multiplied by the flow through the system. These three quantities are, respectively, N, $R+Z$, and X_0. Solving $N = (R+Z)X_0$ for the response time, we obtain the response-time formula: $R = N/X_0 - Z$. In the extreme case of response time almost zero --- because all the servers are ultra-fast --- the system throughput would be $X_0 = N/Z$ because transactions are flowing at the rate individual users complete their thinking intervals.

These formulas are sufficient to answer the throughput and response-time calculation questions posed earlier for the airline reservation network. We are given that each transaction generates an average of 10 directory-disk requests, and so V_i= 10 for the server represented by the directory disk. The mean service time at the directory disk is five milliseconds, so that $S_i = 0.005$ second. The directory disk's utilization is 80%: $U_i = 0.8$. Combining the forced-flow law and the utilization law, we have for total system throughput:

$$X_0 = U_i/V_iS_i = 0.8/(10*0.005) = 16 \text{ transactions per second}$$

Thus the entire airline reservation system is processing 57,600 transactions per hour. The response time experienced by any one of the 1000 agents is:

$$R = N/X_0\text{-}Z = (1000/16)\text{-}60 = 2.5 \text{ seconds.}$$

Bottlenecks

Every network has a bottleneck: A server that is slower than all the others and limits the overall throughput. Speeding up a bottleneck can yield a significant improvement in system throughput. Speeding up a nonbottleneck may hardly affect throughput. Our performance predictions will depend our knowledge of the bottlenecks.

Finding the bottleneck is easy. Suppose that the visit ratios and mean service times do not vary with N. Each server generates a potential bottleneck that would limit the system throughput to $1/V_iS_i$ and would give a lower bound to the response time of NV_iS_i - Z. Obviously, the server with the largest value of V_iS_i gives the tightest limit and is the real bottleneck. The products V_iS_i are sufficient to determine lower bounds on the response time as a function of N (Fig. 5).

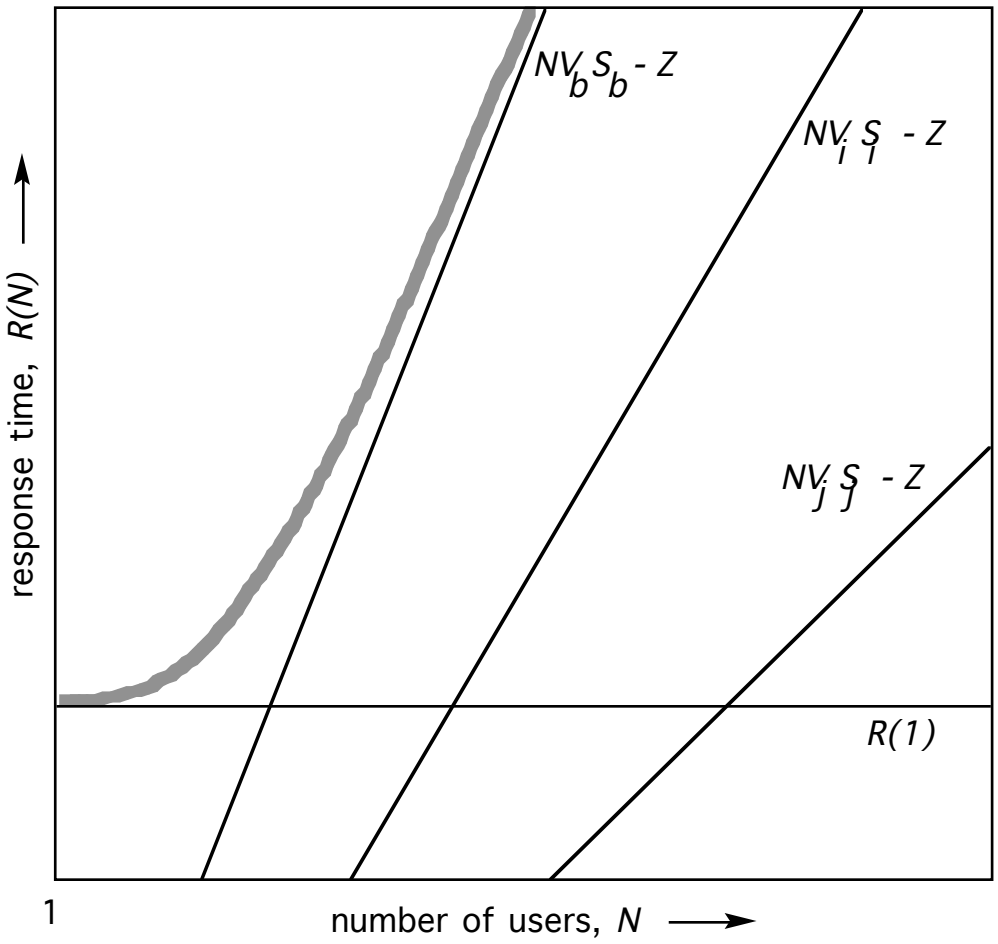

Figure 5. Bottleneck analysis shows how the response time changes as a function of N. When $N = 1$, the single user's transactions encounter no queueing delays from other transactions, whence $R(1) = V_1S_1 + \ldots + V_KS_K$, where K is the number of servers. Combining the utilization and forced-flow laws, $X_0 = X_i/V_i = U_i/V_iS_i < 1/V_iS_i$ because $U_i < 1$. Thus $R(N) > NV_iS_i\text{-}Z$ for all i. Each of the lines defined by these relations is a potential asymptote for $R(N)$ with large N. The actual asymptote is determined by the largest of the potential asymptotes. Taking server b (for bottleneck) to be the one with the largest V_iS_i, we have $R(N) > NV_bS_b\text{-}Z$. The bottleneck analysis assumes that the products V_iS_i do not vary with N.

The operational laws coupled with bottleneck analysis offer a simple but powerful method for performance analysis. For systems whose visit ratios and service times do not vary with overall load, the products V_iS_i ---the total service time requirement for each server---are sufficient to answer these questions.

Changing the Configuration

Consider the two prediction questions we asked at the beginning. They ask about the future effect of reducing directory-disk visit ratio or doubling the number of agents. The operational laws, which deal only with relations among quantities observed in a single measurement period, are not sufficient for making predictions. We must introduce additional forecasting assumptions that extrapolate measured parameter values from the past observation period into the future observation period; the laws can then be used to calculate the response time expected in that future period.

The most common type of forecasting assumption is that, unless otherwise specified, the visit ratios V_i, mean service times S_i, think time Z, and overall load N will be the same. When we change the workload or the strategy for processing data, or the electrical, mechanical, parallelism of a server, we alter only the affected parameters.

Our first prediction question asks what happens when the demand for the directory disk is cut in half (its speed is doubled). The answer depends on whether the directory disk is the bottleneck:

1. If another server is the bottleneck, then speeding up the directory disk will not change the limit on system throughput.
2. If the directory disk is initially the bottleneck, but is no longer the bottleneck after its speed is doubled, then the throughput improves to the limit imposed by the second-slowest server.
3. If the directory disk is still the bottleneck after its speed is increased, then the system throughput improves to the new limit imposed by the directory disk.

The limits of cases 2 and 3 may be superseded by a think-time-imposed limit. No matter how fast the servers are, the maximum rate at which users submit transactions is N/Z; therefore, N/Z is also a limit on the system throughput. Cases 2 and 3 cannot improve beyond that limit.

The operational laws can yield nonsense if the bottleneck effects are ignored. For example, if we assume that doubling the directory disk speed will also double the system throughput, we would change the throughput from 16 to 32 in the response time formula and calculate an absurdity:

$$R = N/X_0 - Z = (1000/32) - 60 = -28.75 \text{ seconds},$$

The think-time constraint on throughput is $N/Z = 1000/60 = 16.7$ transactions per second. If we hypothesize that throughput X_0 can be larger, the response time law tells us that $R = N/X_0 - Z$ is less than zero. That is impossible.

All we can say with the given information and the given forecasting assumptions is that halving the demand for the directory disk will reduce the response time from 2.5 seconds to some small but still nonzero value. If the 2.5-second

response time is acceptable, then this proposed change in directory search strategy would not be cost effective.

Consider the second configuration question: What happens to the response time if the number of agents is doubled? Again, we are limited by the lack of knowledge of the other disks. If the directory disk is the bottleneck, then doubling the number of agents is likely to increase its utilization to 100%, giving a saturation value of throughput:

$$X_0 = 1/V_iS_i = 1/(10*0.005) = 20 \text{ transactions per second}$$

with corresponding response time,

$$R = N/X_0\text{-}Z = (2000/20)\text{-}60 = 40 \text{ seconds}$$

If the directory disk is not the bottleneck, then some other server will have a smaller saturation throughput, which forces response time to be longer than 40 seconds. Thus, doubling the number of agents will produce a response time that is likely to be unacceptably high.

Computational Algorithms

The simple methods described above cannot answer the more complex question of how throughput and response vary with the load *N* on the system. These questions can be answered with very simple algorithms that can be programmed easily on a spreadsheet or hand-held calculator.

The networks-of-queue model was first proposed by Jackson in 1957 *(3)*, and mathematical expressions for its steady-state probabilities were presented by Gordon and Newell in 1967 *(4)*. Their expressions were, unfortunately, exceedingly complex. To calculate a simple quantity such as central processing unit (CPU) utilization required summations over enormous state spaces whose size grew exponentially with *N*. The computational algorithms for these models thus seemed to be intractable. Consequently, these models were not taken seriously, even though a few sporadic experimental studies showed they worked well. In 1973, Jeffrey Buzen presented a breakthrough: an algorithm that computed performance metrics from the Gordon-Newell model in time $O(N^2)$ *(5)*. Buzen's algorithm led to many studies that validated the models and extended them to many new cases and systems. Performance evaluation became the focus of a large and flourishing industry. One of the important extensions was Mean Value Analysis from Martin Reiser and Steve Lavenberg in 1980 *(6)*, which eventually became the industry standard.

The Mean Value Algorithm is so named because it computes a set of means for a closed network (fixed load *N*) --- the response times $R_i(N)$, queue lengths $Q_i(N)$, throughputs $X_i(N)$, and the system throughput $X(N)$ and response time $R(N)$. (Note that we have dropped the subscript "0" from the system throughput

notation.) It does this iteratively for N=1,2,3,… starting with the observation that the queue lengths are all 0 when N=0.

The box below summaries the equations that the algorithm uses to obtain the mean values for load N once the mean values for load $N-1$ have been calculated. Following is an explanation for each of the equations.

Mean Value Equations

(1)	$R_i(N) = S_i(1 + Q_i(N-1))$	for all i
(2)	$R(N) = \sum_{i=1}^{K} V_i R_i(N)$	
(3)	$X(N) = \frac{N}{R(N) + Z}$	
(4)	$Q_i(N) = X(N) V_i R_i(N)$	for all i

(1) When a job arrives at server i, it waits in the queue. That queue's length just before the arrival is approximated as the overall mean queue length when the arriving job is not present (load $N-1$). Just after the arrival, that queue's length is one larger. The arriving job's response time is one service time S_i for each job in the queue just after its arrival. (We will discuss the accuracy of this approximation shortly.)

(2) The overall response time is the sum of all per-visit server response times over all visits. This sum is actually an operational law. Multiply both sides by $X(N)$, apply Little's law to reduce the left side to N, and apply both the forced-flow law and Little's law to convert each term $X(N)V_iR_i(N)$ to $X_i(N)R_i(N)$ and then to the mean queue $Q_i(N)$. The result is the identity that N is the sum of the queue lengths of all the servers.

(3) This equation is the response-time law solved for $X(N)$.

(4) This equation is Little's law applied at each server.

Figure 6 is a very simple version of the airline reservation system. Figure 7 illustrates what the Mean Value Algorithm yields when applied to this model. It is easy to answer the two prediction questions simply by altering the parameters to their new values and applying the algorithm.

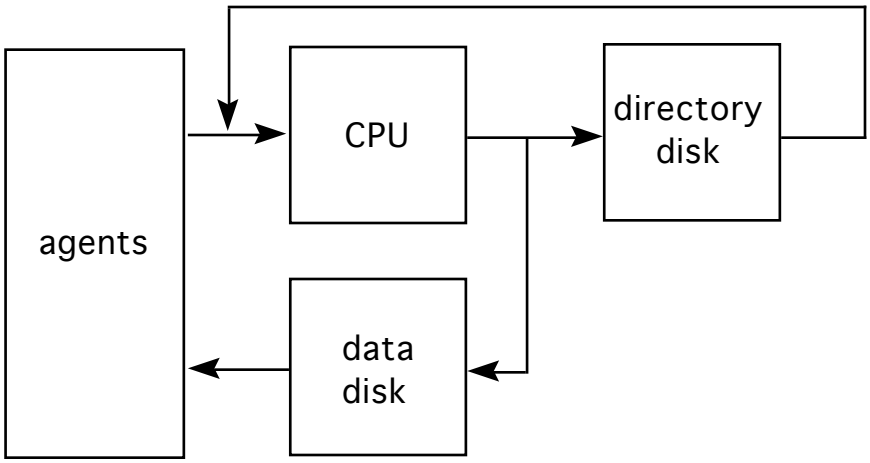

Figure 6. The hypothetical airline reservation system serves as an example of a computer network subject to mathematical performance analysis. In the initial configuration, 1000 agents access a database at the airline's central computing facility. Each agent thinks an average of 60 seconds between transactions. A typical transaction requires 10 lookups on the directory disk to locate the information requested, and then one lookup on the data disk to deliver the result. Each of these 11 disk accesses also requires service from the CPU. The total CPU time of a transaction averages 50 milliseconds. The directory disk service time is 5 milliseconds and the data disk service time is 60.7 milliseconds.

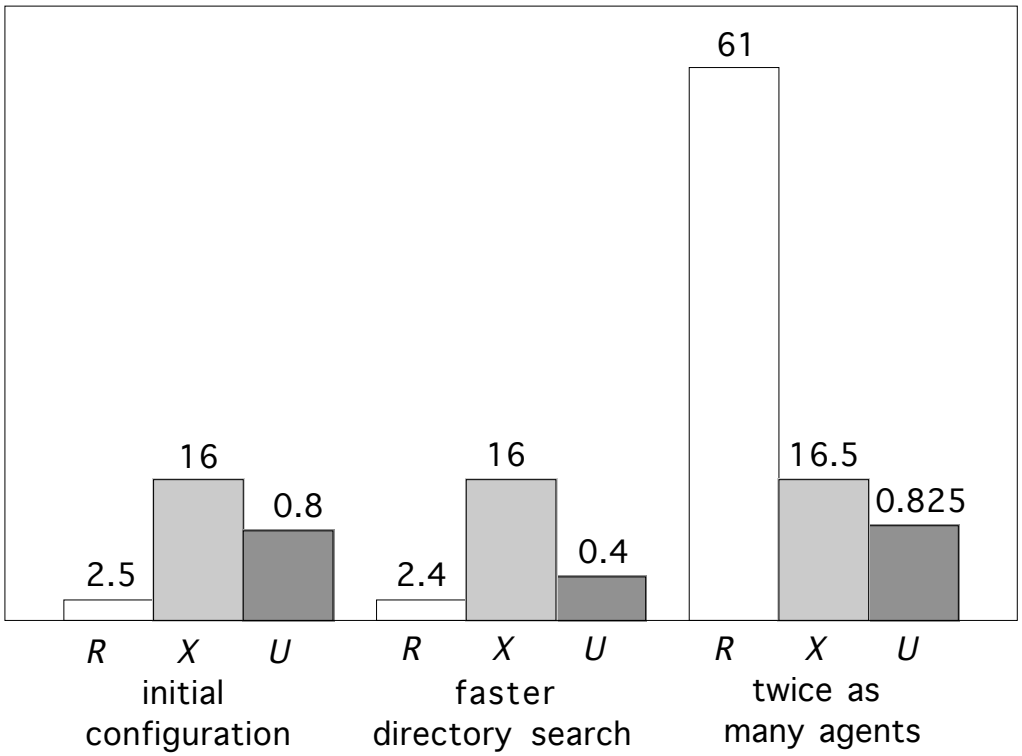

Figure 7. The bar graphs show the results of three network analyses. The original system has an average response time (R) of 2.5 seconds and a throughput (X) of 16 transactions per second; the directory, which seems likely to be the bottleneck, has a utilization (U) of 0.8. If the directory accesses are halved, the utilization of the directory disk falls to 0.4, but the improvement in response time is imperceptible. If the number of agents is doubled, then the response time jumps to 61 seconds.

The Random Arrival Assumption

Our explanation for the Mean Value equations shows that only the first equation contains an approximation; the other three equations are operational laws. Let us consider the nature of this approximation. Whatever errors exist between values calculated from the model and values measured in a system develop from this approximation.

The approximation has two parts: (1) the arriving job observes the same queue length as a random outside observer would with one less job in the system, and (2) all jobs in the queue require the average service time to complete. These assumptions are not necessarily good assumptions. Here's why.

In violation of part 1, the system may have a scheduling algorithm that admits new jobs to active status only when another job leaves. It synchronizes job arrivals with departures. In this case the arrivals do not act as random outside

observers. The arriving job may observe a queue that is shorter than what the random outside observer sees.

In violation of part 2, when a job arrives at a busy server, there is already a job in progress when it arrives. We know from queueing theory that the expected time until the job-in-progress completes is equal to S_i only if the distribution of service times is exponential. If the service times have a long-tail distribution (not unusual), then the expected time until the job-in-progress completes may be considerably larger than S_i. Thus, the assumption that every job in queue needs S_i time to complete may not hold for servers with long-tailed distributions. The arriving job may observe a response time that is longer than the approximation implies.

In both cases, Mean Value Equation (1) will underestimate the true response time. The underestimate may be considerable for long-tailed service distributions.

Few real systems exactly satisfy the assumption behind Equation (1). Yet extensive experiment studies have demonstrated that the models based on it are nonetheless robust: It is almost always possible to construct a model whose estimates of throughput and utilization are within 5% of the true values and whose estimates of response time are within 25% of the true values.

This is quite a remarkable track record for such a simple algorithm.

Extending the Mean Value Equations

The Mean Value Equation (1) is not the only way to approximate the response time. Yon Bard introduced another approximation in consultation with Paul Schweitzer in 1979 *(7)*. The idea was to approximate the mean queue observed by the arriving job as a simple downward proration of the current mean queue length:

$$Q_i(N-1) = \frac{N-1}{N} Q_i(N)$$

After this subsitution, all the mean values mentioned in the equations are functions only of the load N, which can be dropped as an explicit parameter. The result is the simplified equations in the box below.

Bard-Schwetizer Equations for load *N*

(5)	$R_i = S_i(1 + \frac{N-1}{N} Q_i)$	for all i
(6)	$R = \sum_{i=1}^{K} V_i R_i$	
(7)	$X = \frac{N}{R+Z}$	
(8)	$Q_i = XV_iR_i$	for all i

For a given N, these equations are solved iteratively. The algorithm starts with any guess for the mean queue lengths, for example all 1. It then cycles through the equations (5)-(8), which produces successive new guesses for mean queue lengths. The sequence of guesses converges to a set of values that solve the equations. Those mean values are the estimates of response time, throughput, and queue lengths for the system at load N.

Experimental validations have confirmed that the Bard-Schweitzer equations give good approximations in practice, usually with the same errors as the original Mean Value equations.

Another approximation addresses one problem mentioned earlier, that the actual response time is much larger than the Mean Value Equation (1) assumes for long-tailed service distributions. It borrows from queueing theory the Pollaczek-Khintchine formula, which says

$$R_i = S_i\left(1 + \frac{U_i}{1-U_i}\frac{1+C_i^2}{2}\right)$$

where C_i is the coefficient of variation, namely the ratio of service time standard deviation to mean. This formula can replace Equation (5) in the Bard-Schweizer equations, with $U_i=X_iS_i$,. Exponential service time distributions have $C_i=1$, simplifying the formula to $R_i=S_i/(1-U_i)$ for those servers.

Operational Analysis

The discussion above is couched in operational terms: All parameters are taken directly from measured data, and all computed metrics represent measured metrics. For this reason, the analytic approach outlined above is called operational analysis. It begins with the laws and relationships among quantities observable in a system over a time period. It uses these laws to determine the limits bottlenecks impose on throughput and response time. With the additional

assumptions of flow balance and random arrivals, it leads to the Mean Value Equations for calculating the throughput, response times, and mean queues. Operational analysis allows the measurement of errors caused by assumptions such as flow balance or random arrivals.

Traditional queueing theory assumes that stochastic (random) processes govern the performance quantities. Its more powerful methods allow calculation of metrics based on entire service distributions, not just the means. Many steady-state limit theorems of queueing theory turn into operational laws or formulas that hold for flow-balanced networks.

Operational analysis is more intuitive and easier to understand in the most common performance evaluation cases than traditional queueing theory. It gives more credibility to performance models because its assumptions are verifiable. It is commonly used in performance evaluation textbooks to introduce the basic ideas of queueing theory for computing systems and networks *(8)*. Operational analysis, however, is not a replacement for traditional queueing theory.

The genesis of the operational interpretation was in the mid-1970s, when performance analysts were discovering that the formulas of Markovian queueing systems worked very well to predict utilizations, throughputs, and response times in real networks of computers, even though the Markovian assumptions themselves were grossly violated. Jeffrey Buzen proposed the operational hypothesis: Many traditional steady-state queueing formulas are also relations among observable quantities under simple, general conditions *(9)*. This hypothesis has been substantiated in practice and has underpinned many computer programs that accurately calculate performance measures for network-of-server models of computer systems, computer networks, and manufacturing lines.

References

1. P. J. Denning and J. P. Buzen, Operational analysis of queueing networks, ACM *Comput. Surv.* **10**(3): 225-261, 1978.

2. E. D. Lazowksa, J. Zahorjan, G. S. Graham, and K. C. Sevcik, *Quantitative System Performance*, Upper Saddle River, NJ: Prentice-Hall, 1984.

3. J. R. Jackson, Networks of waiting lines, *Operations Res.* **5**: 518-521, 1957.

4. W. J. Gordon and G. F. Newell, Closed queueing systems with exponential servers, *Operations Res.* **15**: 254-256, 1967.

5. J. P. Buzen, Computational algorithms for closed queueing networks with exponential servers. *ACM Commun*. **16** (9): 527-531, 1973.

6. M. Reiser and S. Lavenberg, Mean value analysis of closed multichain queueing networks, *J. ACM* **27**: 313-322, 1980.

7. Y. Bard, Some extensions to multiclass queueing network analysis. Proc. of the Fourth International Symposium on Computer Performance Modeling, Measurement, and Evaluation (H. Beilner and E. Gelenbe, eds.) Amsterdam: North-Holland, 1979.

8. D. Menascé, V. Almeida, and L. Dowdy, *Capacity Planning and Performance Modeling*, Upper Saddle River, NJ: Prentice-Hall, 1994.

9. J. P. Buzen, Operational Analysis: The key to the new generation of performance prediction tools. *Proc. IEEE COMPCON* **76**, Washington, DC: 166-171, 1976.

Further Reading

K. C. Sevcik and I. Mitrani, The distribution of queueing network states at input and output instants. *J. ACM* **28**: 358-371, 1981.