
QUEUEING IN NETWORKS OF COMPUTERS

Peter J. Denning

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 1,000 agents around the country to make reservations from their workstations. There is a "disk farm"---a large collection of magnetic-disk storage devices---in New York that contains all the records of flights, routes, and reservations. On average, each agent issues a transaction against this database once every 60 seconds. One of the disks contains a directory that is consulted during every transaction to locate other disks containing 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 percent of the time.

How many transactions per hour are serviced nationwide on this system? What is the average response time experienced by an agent in Los Angeles? What would happen to the response time if a new method of storing the directory reduced access to five per transaction? What would happen to the response time if the number of agents were doubled?

These are typical questions relating to the capacity of a network of computers to complete the work requested of it. Most people think that the answers cannot be calculated 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 come as a surprise, therefore, that the first two questions---concerning throughput and response time---can be answered precisely from the information given. For the changes of configuration proposed in the third and fourth questions, reasonable estimates of system behavior can be made from the available information and a few plausible assumptions.

Servers and Transactions

A computer network is composed of a number 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 specifying tasks of the type that the receiving server is designed 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 message to another server, requesting that it 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 (*Figure 1*). Similarly, by measuring the

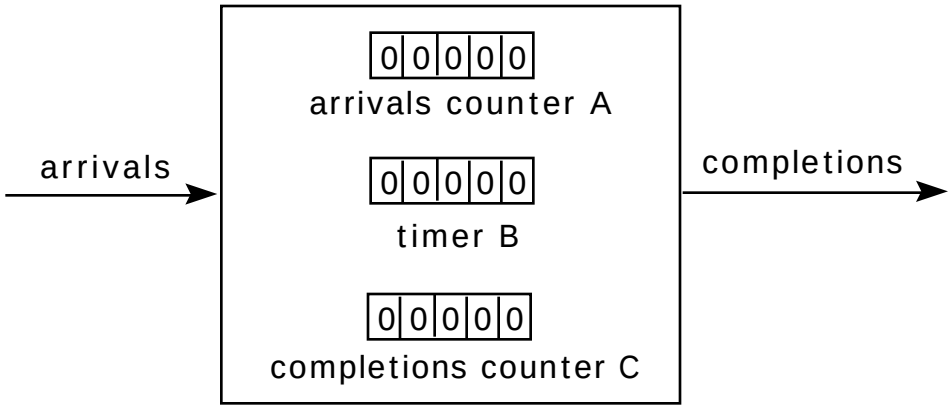

Figure 1. Task-processing server is the basic element of a network of computers. Over an observation period of length *T*, the counter *A* registers the number of tasks arriving at the server, the counter *C* records the number of tasks completed, and the timer *B* measures the total busy time (the time 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 $S = B/C$. Because $B/T = (C/T)(B/C)$, we have the utilization law: $U = XS$.

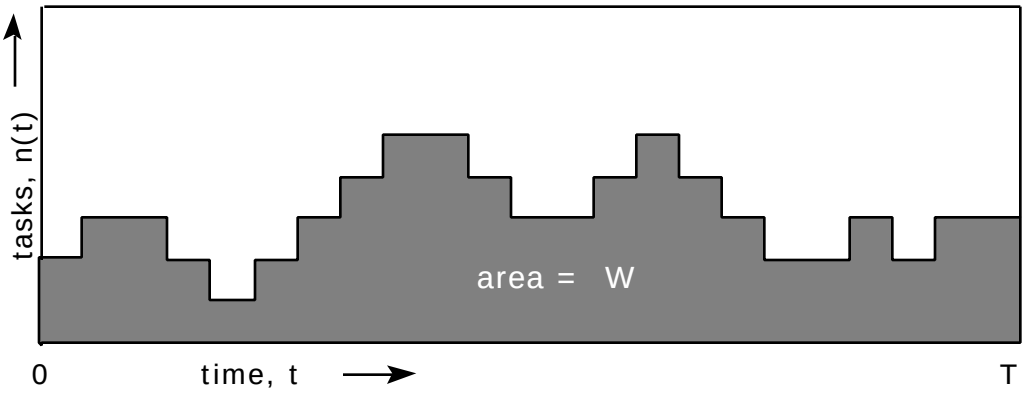

Figure 2. Average response time of a server can be calculated from just a few measurements of the system's performance. Let *n(t)* denote the number of tasks in the server at time *t*. Let *W* denote the area under the graph of *n(t)* in the interval from time 0 to time *T*; *W* is the number of task-seconds 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*. Because *W/T = (C/T)(W/C)*, we have 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.

"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 $Q = RX$, known as Little's Law (*Figure 2*).

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 making 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 (*Figure 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.

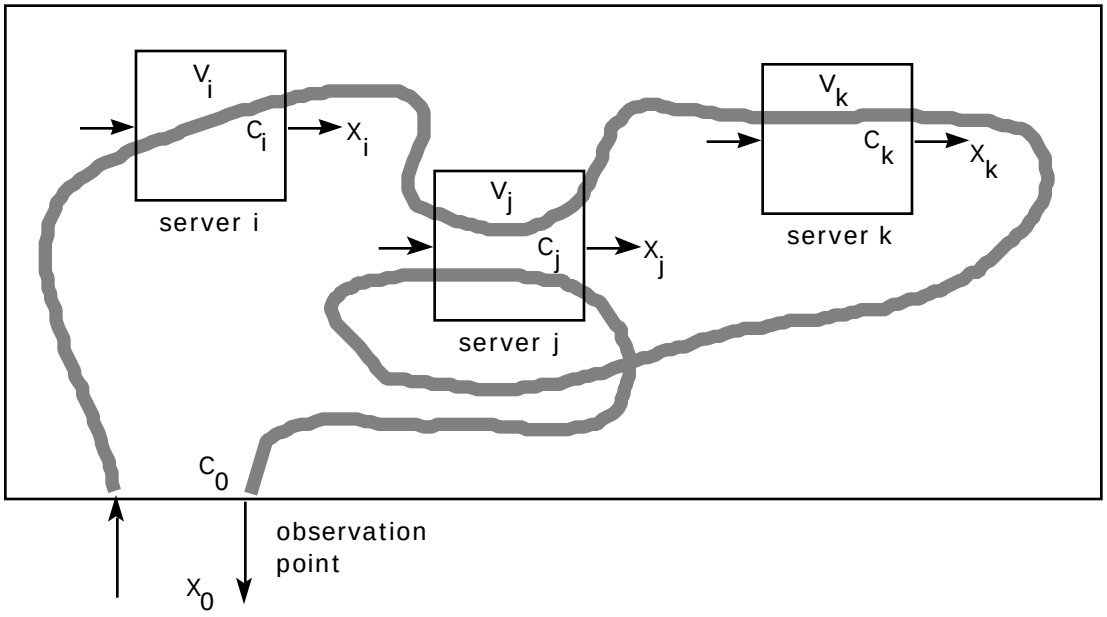

Figure 3. Flow of transactions through a network of servers can be calculated from a few selected measurements of performance. Over an observation period T,C_{ij} transactions are completed by the system. The average number of tasks per transaction for server i is $V_i = C_i/C_0$; V_i is called the visit ratio because each task is regarded as a "visit" by the transaction to the server. Here the transaction visits servers i and k once each and visits server j twice. Because $C/T = (C_i/C_0)(C_0/T)$, we have the forced-flow law: $X_i = V_iX_0$. 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.

In a network, a server's output is a portion of the input to another server or of the output of the system. It simplifies an analysis to assume that the input and output flows of a server are identical and can be called the throughput---a condition known as flow balance. The definitions do not imply flow balance. In most real systems there is a bound 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 to this bound, the error introduced by assuming flow balance will be negligible. For this reason, flow balance does not generally introduce much error for practical systems.

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 $R = N/X_0-Z$ (*Figure 4*). This formula is exact for flow balance.

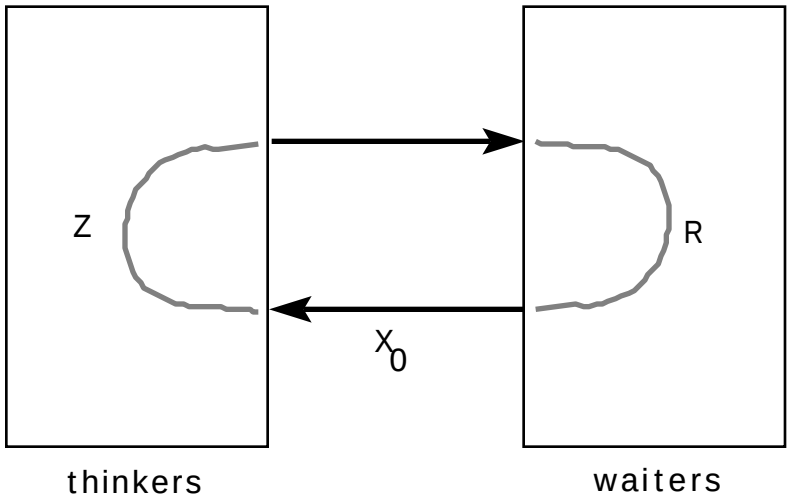

Figure 4. Users of a transaction system alternate between periods of "thinking" and periods of "waiting" for a response from the system. The total number of users---thinkers and waiters--- is *N*. The average response time per transaction is *R* and the average thinking time is *Z*. Little's law says that the mean number of active users in an entire system is equal to the mean response time of the system multiplied by the flow through the system. These three quantities are, respectively, *N*, *R*+*Z*, and X_0 Solving for the response time---or in other words the average period spent waiting---we obtain the response-time formula: $R = N/X_0 - Z$. Since the system includes a fixed number of thinking and waiting users, this formulation assumes one system arrival for each system completion (flow balance).

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

$X_0 = U_i/V_iS_i = 0.8/(10\times0.005) = 16$ 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 1,000 agents is:

$R = M/X_0 - Z = 1000/16 - 60 = 2.5$ seconds.

Changing the Configuration

Consider the next two configuration questions. They ask for grounded speculations about response time in a future measurement period having different conditions---for example, the directory-disk visit ratio is reduced, or the number of agents is increased. Since operational laws deal only with relations among quantities observed in a past measurement period, they 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.

One common type of forecasting assumption is that, unless otherwise specified, the demands placed on the various servers, V_i will be the same in the future period as they were in the measurement period. Similarly, unless otherwise specified, the mean service times, S_i which depend primarily on mechanical and electrical properties of devices, will be the same. The utilizations, throughputs, and response times will change when any of these parameters changes.

The first configuration question asks what happens if some disk other than the directory disk is the bottleneck of the system; most of the transactions are queued there, and its utilization is near 100 percent. Under these conditions reducing the demand for the directory disk will have only a negligible effect on the utilization and throughput of the bottleneck disk; the forced-flow law tells us that the overall throughput and response time of the network will therefore be unchanged.

At the other extreme, the directory disk is the bottleneck of the system; halving the demand on it will double system throughput. For the numbers given above, the response-time formula yields a calculated response time of -28.75 seconds. The obvious absurdity of a negative response time---signifying that answers are received before questions are asked---indicates that the directory disk cannot be the bottleneck after demand on it is reduced by half, even if it was the bottleneck originally. 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 and nonnegative value. If the 2.5-second response time is acceptable, 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 percent, giving a saturation value of throughput:

$X_0 = 1/V_iS_i = 1/(10\times0.005) = 20$ transactions per second

With the response-time formula, these values yield:

$R = N/X_0 - Z = 2000/20 - 60 = 40$ seconds

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

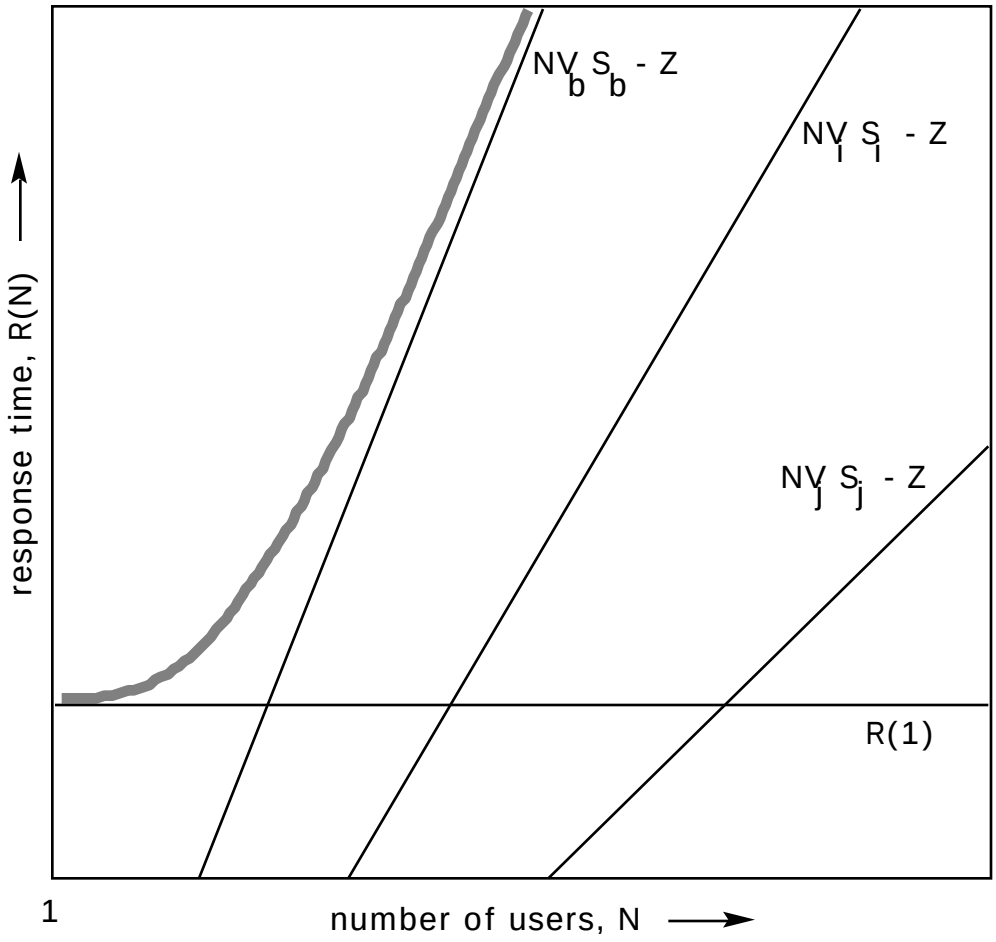

Figure 5. Bottleneck analysis shows how the response time changes as a function of *N*. When *N*=1, the single user's transactions encounter no queueing delays from other transactions, whence $R(1) = V_1S_1 + \ldots + V_KS_K$, where *K* is the number of servers. Combining the utilization and forced-flow laws, $X_0 = X_i/V_i = U_i/V_iS_i < 1/V_iS_i$ since $U_i < 1$. Thus $R(N) > NV_iS_i - Z$ 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 $R(N) > NV_bS_b - Z$. The bottleneck analysis assumes that the products V_iS_i do not vary with *N*.

This example illustrates that bottleneck analysis is a recurrent theme in forecasts of throughput and response time. Suppose that the visit ratios and mean service times are known for all the servers and do not vary with N. Each server generates a potential bottleneck that would limit the system throughput to $1 / V_i S_i$ and would give a lower bound to the response time of $N V_i S_i - Z$. Obviously the server with the largest value of $V_i S_i$ gives the least upper bound on the throughput and is the real bottleneck. The products $V_i S_i$ are sufficient to determine lower bounds on the response time as a function of N (*Figure 5*).

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 $V_i S_i$ ---the total service time requirement for each server---are sufficient to answer these questions. The methods can be extended to yield efficient algorithms for computing throughput, response time, and mean queue length at every server as a function of the load N on the system (*1*,*2*).

Operational analysis is not a replacement for traditional queueing theory; it is a reinterpretation for the common case of measured data. Many of the steady-state limit theorems of queueing theory turn into operational laws or formulas that hold for flow-balanced networks.

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 were grossly violated. Jeffrey P. Buzen proposed the operational hypothesis: Many of the traditional steady-state queueing formulas are also relations among observable quantities under simple and general conditions (*1*). This hypothesis has been substantiated in practice and has become the underpinning for a large number of computer programs that calculate performance measures for networks of servers ranging from computers to manufacturing lines.

Computing with Models

The networks discussed above can be viewed as a collection of users and servers. The users generate transactions, which are processed by various servers, until ultimately a response is returned to the user. The two primary performance metrics, throughput (X) and response time (R) obey certain operational laws that allow them to be calculated from visit ratios (V_i), service times (S_i), and total number of users (N). When we start asking “what-if” questions about future configurations these laws give useful approximations when combined with bottleneck analysis. Through these simple methods, “back of the envelope” calculations can quickly reveal the effects of distant servers on local throughput and response time.

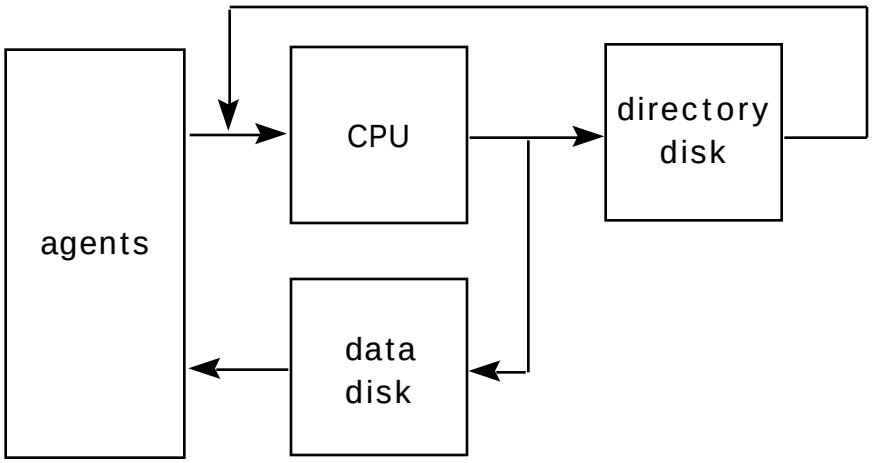

Figure 6. Hypothetical airline reservation system serves as an example of a computer network subject to mathematical performance analysis. In the initial configuration 1,000 agents access a database at the airline's central computing facility. Each agent thinks an average of 60 seconds between transactions. 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 central processing unit (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.

Suppose one is willing to move beyond the backs of envelopes to programmable calculators or spreadsheets. Are there simple extensions to the basic results that will allow more accurate calculation of response time as a function of the load for given values of the visit ratios and mean service times? The answer is yes.

To keep the argument simple, let us confine our attention to a straightforward but useful case. Assume that the basic server parameters do not change with the total load on the system N. Further assume that the system is closed (N is fixed) and that there is an observation point at which the network throughput X and response time R are defined.

To calculate the total system response time, we need to know the response time of each server. Since a transaction visits each server V_i times and experiences a mean response time R_i on each visit, we can perform the calculation by means of the operational law

$$R = V_1 R_1 + \ldots + V_K R_K$$

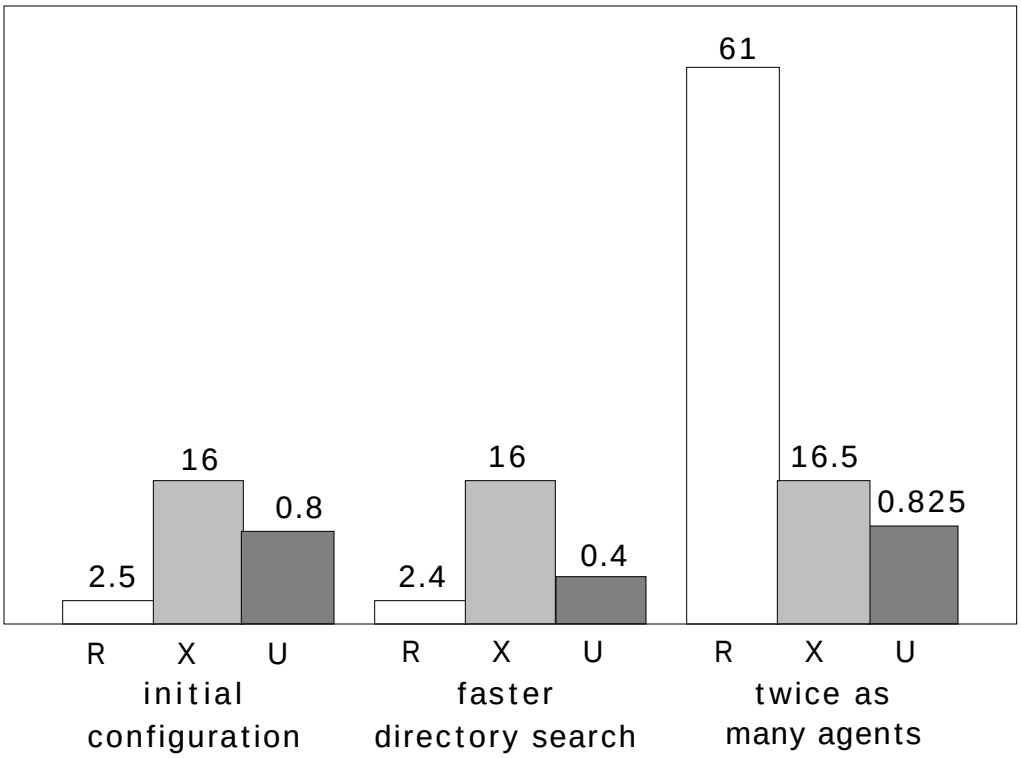

Figure 7. The bar graphs show the results of three network analyses. The original system has an average response time (*R*) of 2.5 seconds and a throughput (*X*) of 16 transactions per second; the directory, which seems likely to be the bottleneck, has a utilization (*U*) of 0.8. If the directory accesses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

This law can be verified by multiplying both sides by X, using Little's law to reduce the left side to N, using the forced-flow law and Little's law to reduce each term XV_iR_i on the right side to mean queue length Q_i and noting that the result is the identity $N = Q_1 + ... + Q_K$.

Now we must determine the local response time, R_i of an individual server. A starting point for calculating the value is P_i the mean queue length as determined from measurements of the length as each task arrives at server *i*. Since the response time consists of a task's own service time plus the service times of the tasks that precede it in the queue, R_I is given by the formula:

$$R_i = S_i(1 + P_i)$$

Unfortunately, the mean queue length seen at the arrival instants is usually not the same as the queue length averaged over all times (*3*). An example where arrival-time measurements give misleading results is a server that is in low demand. Each arriving transaction finds the queue empty, and service is always completed before the next task comes along. In this case the mean queue length seen by arrivals is 0, whereas the mean length determined by an outside observer---by someone who records measurements at all times---would be a number between 0 and 1.

Observers of Queues

What is the relation between the queue lengths seen by arriving tasks and by outside observers? The simplest possible relation arises when we convert the arriving task into an external observer, simply by excluding that task from the system under observation. In other words, we divide the network into two parts: the arriving transaction, which has the role of observer, and the remainder of the tasks and servers. When the network is partitioned in this way, the mean queue length observed by arrivals is the same as the time-averaged mean queue length would be if there were one fewer task in the system. We write this relation as

$$P_i(N) = Q_i(N-1)$$

Let us further postulate a linear relationship between the mean queue lengths at loads N and N-1:

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

Now we can make use of this result to define a better expression for the local response time:

$$R_i = S_i\left(1 + \frac{N-1}{N} Q_i\right)$$

When augmented with the forced-flow law and Little's law, this last equation yields an iterative scheme for computing response time, the throughput, and the mean queue lengths when given values of the parameters V_i, S_i, and N. The equations and the iterative method were first proposed by Yon Bard and Paul Schweitzer (*4*).

The first step in the iterative procedure is to estimate the mean queue length Q_i for each server *i*. The estimates do not have to be highly accurate; one simple strategy that yields acceptable results is to apportion the total number of tasks N equally among the servers. The initial Q_i estimates are then employed to calculate a response time R_i for each server, and the R_i values (along with the

visit ratios V_i) determine the overall response time R. N and R together fix the throughput of the system X. Finally, from X, V_i and R_i one can make refined estimates of the mean queue lengths Q_i. This procedure is repeated until the difference between successive estimates is less than some error bound. Fifty or a hundred iterations may be needed to reduce the error to 0.0001.

The foregoing analysis relies on a number of assumptions. The parameters V_i and S_i are assumed to be independent of the load N; the mean queue length at load N is assumed to be linearly related to the mean length at load N-1; and arriving tasks are assumed to act as outside observers of the queues at their servers. The first of these assumptions can be relaxed to allow the visit ratios to depend on total load N and to allow the service times to depend on the local queue lengths. It is further possible to distinguish separate classes of transactions, each with its own set of parameters. Algorithms for computing the mean queue lengths, throughputs, and response times are given for all these cases in the books by Lazowska et al. (*2*) and Menascé et al. (*6*).

Relaxing the second assumption---that network performance is a linear function of load---turns the iterative algorithm into a recursion on N. Given a set of values of Q_i at load N-1, a single pass through the recursive algorithm produces values of Q_i at load N. This approach, called the mean-value iteration, was first proposed by Martin Reiser and Stephen Lavenberg (*5*).

The third assumption---that an arriving task can serve as an observer of queue length---cannot be relaxed. It is the key assumption that enables queueing networks to be analyzed by simple computational algorithms. Few real systems satisfy the assumption, but the models based on it are nonetheless quite robust: It is almost always possible to construct a model whose estimates of throughput and utilization are within 5 percent of the true values and whose estimates of response time are within 25 percent of the true values.

References

1. Peter J. Denning and Jeffrey P. Buzen. 1978. Operational analysis of queueing networks. ACM *Computing Surveys 10*, 3 (September): 225-261.

2. Edward D. Lazowksa, John Zahorjan, G. Scott Graham, and Kenneth C. Sevcik. 1984. *Quantitative System Performance*. Prentice-Hall.

3. Kenneth C. Sevcik and Isi Mitrani. 1981. The distribution of queueing network states at input and output instants. *Journal of the ACM 28*: 358-371.

4. Yon Bard. 1979. Some extensions to multiclass queueing network analysis. Proceedings of the Fourth International Symposium on Computer

Performance Modeling, Measurement, and Evaluation (H. Beilner and E. Gelenbe, eds.) Amsterdam: North-Holland.

5. Martin Reiser and Stephen Lavenberg. 1980. Mean value analysis of closed multichain queueing networks. *Journal of the ACM 27*: 313-322.

6. Daniel Menascé, Virgilio Almeida, and Larry Dowdy. 1994. *Capacity Planning and Performance Modeling*. Prentice-Hall.