MANAGEMENT SCIENCE
Vol. 29, No. 10, October 1983
Printed in U.S.A.

A FLUID FLOW MODEL OF NETWORKS OF QUEUES*

JAMES S. VANDERGRAFT†

This paper describes a new technique for modeling flow through a network of queues. The advantages of this method over many discrete event simulations or queueing theory techniques include simplicity, low cost and portability. The paper describes the kinds of processes to which the technique can be applied, and the characteristics of the process that can be determined by the resulting model. An example of claims processing in a Social Security Administration's District Office is given. The basis for the technique is to model the information flow by a fluid flow, and then use standard engineering ideas to describe the fluid flow by a system of ordinary differential equations. The system of equations is solved by well-known numerical methods.
(NETWORKS; QUEUEING THEORY; SIMULATION)

1. Introduction

In this paper, a novel technique for modeling flow through a network of queues is described. The basis for the technique is to think of the flow as a fluid flow and then use a standard engineering idea to describe the fluid flow by a set of differential equations. The differential equations are solved by well-known numerical methods. The advantages of this approach include simplicity, low cost and portability. The programs needed to implement the technique are fairly simple and can be written in any high level language, such as FORTRAN. The running time for the resulting program is very short. Because of the simplicity of the programs, it is easy to tailor them to particular situations or to "fine tune" them to obtain a more accurate simulation. It is important to stress, however, that unlike standard queueing theory techniques, the method described here is deterministic. Thus, the effects of random arrivals, or of variations in service times, cannot be studied directly.

In §2, the kind of problem to which this technique can be applied is illustrated. An example of claims processing in a Social Security Administration District Office is given in §3.

The mathematical basis of the technique is developed in §4, and some remarks on implementation details are explained in §5. Finally, in §6, this new technique is compared to more standard methods for modeling this kind of flow problem.

2. An Overview of Applicable Problems

This modeling technique can be applied to any processing situation that can be described by a set of flow diagrams which show the order in which the processing is done. Figure 1 gives an example of a typical flow diagram.
The boxes in Figure 1 can represent:

- work stations,
- computer system components,
- other processing units.

(A diagram such as in Figure 1 is often referred to as a "network of queues". However, it is easier to describe the modeling technique by using flow theory terminology.)

*Accepted by George S. Fishman; received February 10, 1981. This paper has been with the author 4 months for 1 revision.
†Automated Sciences Group, Inc., Silver Spring, Maryland.

0025-1909/83/2910/1198$01.25

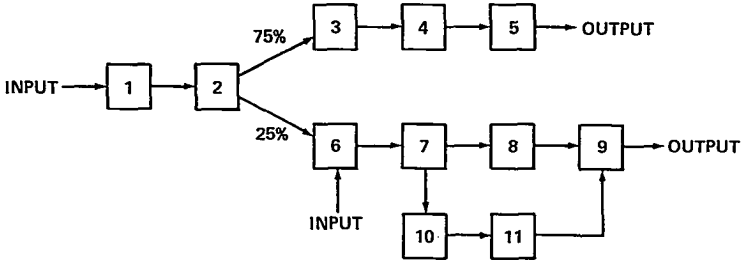

FIGURE 1. A Typical Flow Diagram.

The links between the boxes show how information flows from one box to another. To allow for slight variations in the processing, it is possible to attach percentages to some of the links to show how much of the flow goes along any of several different paths.

The only information that must be known about each box is its *processing rate*; that is, the average number of items that can be processed at this box in a fixed time interval. This rate need not be constant, but may depend, for example, on

- time of day,
- workload,
- availability of resources
  - — number of employees,
  - — computer memory,
  - — processing units.

The *inputs* noted in Figure 1 may be either *incremental* or *bulk*. Incremental arrivals refer to small quantities of items that arrive throughout the time period being modeled. Examples of this type of arrival are:

- clients arriving for service,
- computer demand jobs,
- phone calls.

Bulk arrivals, on the other hand, refer to large quantities of items that arrive only a few times each day. Examples here include:

- mail deliveries,
- batch job submittals.

Whenever the process to be analyzed can be described in the above manner, the technique explained in §4 can be used to simulate it. Characteristics of the process that can be determined by the technique include:

- *Productivity*: The number of items that pass through each box, or the number of items that reach the output points.
- *Backlogs*: The number of items waiting at each box.
- *Utilization of Resources*: The percentage of available resources (people or equipment) being used.
- *Processing Time*: How long it takes for a single item to be completely processed, including waiting time.
- *Delay Times*: How long items wait for processing.

All of these quantities can be computed at selected times during the analysis period, every hour, for example.

Because the programs are inexpensive to run, the analysis can be repeated many times to show the effects of changing parts of the processing procedure.

If the boxes in the flow diagrams represent work stations that are staffed by several workers, then the modeling technique can also be used for staffing analysis. That is,

the minimum number of workers needed to keep work flowing smoothly through the office can be computed directly from the modeling equations. Also, it is possible to compute the number of workers that must be assigned to particular workstations so that the waiting times meet specified standards. For details of these staffing analyses, see [8] and [9].

3. An Example—S.S.A. Claims Processing

The Social Security Administration has a network of District Offices located throughout the country. Each office is set up to process claims that are filed by local citizens. There are over 30 basically different types of claims, and each type is handled differently from the others. There are slight variations among claims of the same type, but generally those variations are few and can be accurately predicted. Thus, the claims processing activities within a particular District Office can be described by a set of flow diagrams; one for each type of claim handled by that office. All of these diagrams are of the type illustrated by Figure 1. The inputs to Box 1 are, typically, claimants who come to the office to be interviewed by a Claims Representative (CR) or a Service Representative (SR). A folder is prepared and passed on to another worker type such as a Claims Development Clerk (CDC), or a Data Review Technician (DRT). Additional information may come by mail (at Box 6 for example) for some of the claims, while other claims proceed directly through a sequence of processing steps to the output stage. The output here represents the complete processing of a claim. Usually, at this stage, the claimant is notified and the folder is stored in a file cabinet.

Thus, each box in the flow diagrams for this application can be thought of as a workstation that is staffed by one or more workers of a particular type: CR, SR, DRT, CDC, TT (teletypist), or ANC (account number clerk). The processing rate associated with each box will depend on how many workers are assigned to the workstation that the box represents. That is, the more workers there are, the more claims that can be processed in, say, one hour. For each workstation, estimates are available for how many items can be processed by *one* worker per hour. If this is multiplied by the number of workers assigned to the workstation, the result is the processing rate for the station.

The number of workers assigned to each station depends on the amount of work, as well as on the number of workers available. As claims begin to flow through the workstations, workers are assigned to process them. As long as there are only a few claims in the office, each claim will receive immediate attention from an available worker. As claims begin to accumulate, however, workers will have to be assigned to workstations depending on the number of claims to be processed at each station. The worker assignment algorithm is discussed in more detail in §4. For now, it suffices to note that the worker assignments change throughout the day; hence, the processing rates also change. Furthermore, this reflects what actually happens in practice.

Programs have been written in standard FORTRAN to simulate the activities in the Patchogue, New York District Office. The simulation begins by assuming there is no work in the office. At 8:00 a.m., claimants begin to arrive for interviews. As they are interviewed, folders are prepared which are later passed on to other workstations for processing. Workers are assigned as needed, or reassigned as work begins to accumulate. At 11:00 a.m., the mail is distributed. Between 12:00 p.m. and 1:00 p.m., half of the workers go to lunch, each for one-half hour, and at 5:00 p.m., all work stops. This is considered to be day zero of the analysis. Day *one* is exactly the same, except that it begins with work left over from the previous day. The analysis can continue for as many days as desired.

The input to the program consists of four tables. A *node table* is used to give information about each workstation; this information includes a workstation identification, worker type, task description, process rate per worker, and external arrival information. The connections between the workstations are described by a *link table* that gives the identifications of the workstations at each end of the link, and the flow percentage through the link. The client arrivals are defined by an *arrival table* that gives, in tabular form, the number of claimants that have come to the office by any time of day. The fourth table is a *worker table* that tells how many of each type of worker is in the office.

Figure 2 shows an example of the output from the model of the Patchogue, N.Y. District Office. Careful validation of these results is difficult because of a lack of accurate data with which to compare the computed values. However, the qualitative characteristics predicted by these results agree with observed activities and conditions in the office.

The programs that produced these results consist of approximately 2,000 FORTRAN statements, not counting comments. Of these, nearly 500 statements constitute the subroutine RKF45 which was copied from [5]. This subroutine solves the system of differential equations that describes the flow through the diagrams. Of the remaining statements, over half are used to accumulate the results in several different ways, and format them for printing. The running time on a DEC 2020 system was approximately ten seconds. This included reading the input tables from disks, setting up the equations, solving the equations for two eight-hour days, and accumulating the results for printing. The flow diagrams used here contained 151 nodes.

The programs allow the user to interactively change some of the input, such as staffing levels, number of arrivals, and processing times, More extensive changes to the office can be simulated by simply editing the node and link tables.

4. Mathematical Details

In this section, the equations that are used to describe the flow of work are derived. The estimates of waiting time and the staffing analysis formulas are also discussed.

4.1. *Continuity Assumptions*

The flow of work through a flow graph such as that given in Figure 1 can be described, approximately, by a system of ordinary differential equations. With the nodes numbered $1, 2, 3, \ldots$, as in Figure 1, let $w_k(t)$ denote the number of work items at node k at time t. The value of $w_k(t)$ is determined by the flow of work into and out of node k. Initially, the values of all w_k are zero, unless there is a backlog of work when the analysis begins. Additional work items enter the system as incremental and bulk arrivals. Let $\Lambda_k(t)$ denote the number of incremental arrivals to node k by time t. For most nodes, $\Lambda_k(t) = 0$, but for nodes such as node 1 in Figure 1, $\Lambda_1(t)$ is an increasing function of t. (Remember that $\Lambda_k(t)$ is the *total* number of arrivals from $t = 0$ until the present time.) In fact, if $\Lambda_k(t)$ is not zero, then it must be a step function, as shown in Figure 3.

The steps (discontinuities) occur whenever a work item arrives. An assumption that must be made in order to analyze the flow by using differential equations is that $\Lambda_k(t)$ is a continuous function. In fact, it is even necessary to assume that Λ_k is a differentiable function. One such differentiable approximation is shown by the dotted curve in Figure 3.

Next, consider the flow out of a typical node. This will depend on the rate at which work is processed at the node. In many situations of interest, this rate is time dependent, hence the notation $\mu_k(t)$ will be used to denote the average number of

SUMMARY OF THE RESULTS FOR THE 1ST DAY

DISTRICT OFFICE SUMMARY

TIME	BACKLOG (WORKHOURS)	PRODUCTIVITY (NUMBER OF ITEMS)	UTILIZATION (% OF WORKERS)	ESTIMATED TOTAL TIME (HOURS)	CLIENT WAITING TIME FOR CR	FOR SR
8:00	34.	0.	71.	4.9	.00	.00
9:00	19.	69.	64.	3.8	.00	.00
10:00	27.	128.	76.	3.6	.28	.00
11:00	74.	175.	96.	9.1	.43	.50
12:00	86.	246.	96.	10.3	.50	.50
1:00	91.	276.	66.	19.3	.50	.50
2:00	93.	334.	95.	12.6	.50	.50
3:00	84.	387.	77.	10.5	.50	.50
4:00	67.	432.	71.	7.5	.50	.50
5:00	50.	467.	70.	6.4	.50	.50
MEAN	63.		78.	8.8		

WORKER SUMMARY

	WORKHOUR BACKLOG/PERCENT UTILIZATION					
TIME	CR	SR	DRT	CDC	TT	ANC
8:00	24.3/ 60.	2.0/ 86.	1.8/100.	1.5/ 75.	4.2/100.	.0/ 20.
9:00	14.4/ 98.	.3/ 43.	.2/ 20.	.6/ 35.	4.0/100.	.1/ 35.
10:00	21.5/100.	.5/ 74.	.4/ 52.	.6/ 41.	3.3/100.	.3/100.
11:00	26.0/100.	18.3/100.	.5/ 61.	22.9/100.	3.0/100.	3.8/100.
12:00	49.0/100.	15.7/100.	.5/ 53.	15.0/100.	3.4/100.	2.8/100.
1:00	57.0/ 84.	14.5/ 67.	.5/ 50.	12.2/ 50.	4.0/ 50.	3.0/ 50.
2:00	69.5/100.	12.1/100.	.4/ 50.	4.5/100.	4.6/100.	2.3/100.
3:00	67.3/100.	9.4/100.	.4/ 49.	.6/ 35.	5.0/100.	1.3/100.
4:00	54.6/100.	5.9/100.	.3/ 36.	.5/ 32.	5.4/100.	.0/ 22.
5:00	41.5/100.	2.4/100.	.2/ 27.	.5/ 33.	5.4/100.	.0/ 20.

FIGURE 2. Example of Output from the Differential Equation Model.

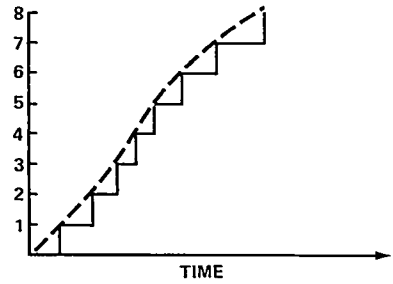

FIGURE 3. A Typical Arrival Function $\Lambda_k(t)$ and a Continuous Approximation to it.

hours to process one item at node k at time t. It must be assumed that $\mu_k(t)$ is continuous, except perhaps at a few times. In practice, this assumption does not seem too unrealistic.

With these two continuity assumptions, it is not difficult to show that the number of items, $w_k(t)$, is also a continuous, and differentiable, function. The idea of using a continuous function to represent quantities that take only integer values is standard practice in mathematical modeling. See, for example, [1] and [2].

4.2. *The Differential Equations*

By using the notation defined in the previous section, the flow through the graph can be described approximately by the set of differential equations

$$\dot{w}_k(t) = \dot{\Lambda}_k(t) + \sum_i b_{ik}\mu_i^{-1}(t) - \mu_k^{-1}(t), \qquad k = 1, 2, \ldots . \tag{4.1}$$

Here, the summation is over all nodes that feed into node k, b_{ik} is the fraction of the flow out of node i that goes to node k as indicated in the flow graph, and the dot represents the time derivative.

Since $\mu_i(t)$ represents the time needed to process one item at node i, it is reasonable to assume that $\mu_i(t) > 0$. Hence, $\mu_i^{-1}(t)$ is defined and in fact is the rate at which items are processed. Thus, (4.1) expresses the fact that the change in the number of items at node k equals the rate of incremental arrivals, plus the rate at which items are fed from other nodes, minus the rate at which items are processed and passed out of node k. The initial conditions for this system are values for $w_k(0)$. If there are no work items at the beginning of the analysis period, then $w_k(0) = 0$. However, if there is a bulk arrival at time 0, or if there is work left over from an earlier time (the previous day, for example), then $w_k(0)$ may be nonzero.

For a bulk arrival to node k, at time t_0 say, the equations are solved from time 0 to time t_0. Then the value of $w_k(t_0)$ is increased by the amount of the bulk arrival, and the solution process is continued, with this new value of $w_k(t_0)$ as an initial condition.

A simpler way to treat incremental arrivals is to let $Z_k(t) = w_k(t) - \Lambda_k(t)$. Then (4.1) can be written also as

$$\dot{Z}_k(t) = \sum_i b_{ik}\mu_i^{-1} - \mu_k^{-1}. \tag{4.2}$$

This avoids the problem of finding an approximation to $\dot{\Lambda}_k$; however, clearly $\Lambda_k(t)$ is still needed so that $w_k(t)$ can be computed from $Z_k(t)$.

As in the situation described in §4.4, the processing times are often nonlinear functions of the backlog w_k; that is

$$\mu_i = \mu_i(t, w_1, w_2, \ldots). \tag{4.3}$$

Thus, (4.1) represents a nonlinear system of differential equations. There will be one equation for each workstation. This system can be solved numerically by any of several efficient and reliable codes. Especially recommended are the subroutine RKF45, as given in [5], or the Adams method described in [7].

4.3. *Productivity*

There are two types of productivity that may be of interest: the number of items handled at a particular workstation, and the number of items that are completed during some period of time.

The number of items processed at workstation k, between time $t = 0$ and time $t = T$ is:

$$P_k(T) = \int_0^T \mu_k^{-1}(t)\,dt. \tag{4.4}$$

This follows from the fact that $\mu_k^{-1}(t)$ is the rate at which items are processed at node k, that is, the number of items processed per hour. Thus, the integral of $\mu_k^{-1}(t)$ is the number of items processed during the time interval of integration. Note that if the workstation is idle, then μ_k is infinite, since an idle station takes infinitely long to do anything. In this case μ_k^{-1} is interpreted as zero, so P_k, as defined by (4.4), is zero.

Values of $P_k(t)$, for selected t's, can be approximated by using the trapezoidal rule to evaluate the integral in (4.4). A more accurate method is to integrate (4.2) to obtain

$$Z_k(T) - Z_k(0) = \sum_i b_{ik} P_i(T) - P_k(T), \qquad k = 1, 2, \ldots . \tag{4.5}$$

This is a system of linear (algebraic) equations for the productivities at each station, which can be easily solved by Gaussian Elimination.

The number of completed work items can be determined by the use of special nodes in the flow diagrams. These nodes, which may be called END nodes, should mark the end-of-processing. Thus, in Figure 1 there should be END nodes following nodes 5 and 9. Associated with END nodes are zero processing rates; that is, $\mu_k^{-1}(t) = 0$, so that any items that enter these nodes stay there. The backlogs at the END nodes will show how many items have entered these nodes since the analysis began.

4.4. *Resource Allocation*

In many applications, the processing times μ_k depend on the availability of resources such as people, computers, memory blocks, etc. To consider this case more carefully, we will assume, for definiteness, that the resource is people.

If the workstations are staffed by workers, then it is natural to define a "per-person" processing time as:

p_k = average time for one worker to process one item at station k.

Then, if there are n_k people assigned to the workstations at time t, the processing time will be

$$\mu_k(t) = p_k / n_k(t). \tag{4.6}$$

The numbers n_k must be determined with some care in order for the model to be realistic. An obvious constraint is that n_k cannot exceed the number of available workers. A slightly more subtle restriction is imposed by the fact that n_k should not exceed some multiple of the number of items $w_k(t)$ at the workstation, where this multiple depends on the level of detail of the work flow descriptions. If, for example, each workstation represents a very simple task to be performed on a work item, then it makes no sense to allow two or more people to share this task on a single item. In this case, it is probably realistic to restrict n_k to be always less than or equal to w_k. We will

impose this restriction throughout the remainder of the discussion. Also, for simplicity, we will assume that any worker can perform any of the tasks represented in the flow diagrams. This is not an essential restriction, as shown by the example in §3.

Thus, we suppose that there are a total of N workers available. Initially, workers can be assigned to workstations on a one-to-one basis with the work items. That is, if $w_k(t) = 3$, then three workers can be assigned to workstation k. However, this simple way of assigning workers must be modified when all available workers have been assigned. Additional work arriving should cause a reassignment of workers, depending upon the distribution of work among the stations. In practice, a manager usually watches for work piling up at certain desks and occasionally reassigns workers or redistributes work to control this buildup of backlogs. This reallocation of workers can be included in the model as follows. Whenever

$$\sum_k w_k \leqslant N \tag{4.7}$$

where the summation is over all workstations, then the number of workers n_k assigned to workstation k is just

$$n_k(t) = w_k(t). \tag{4.8}$$

However, if (4.7) is violated, then (4.8) would assign more workers than are available. In this case, a reasonable replacement for (4.8) is

$$n_k(t) = \frac{w_k(t)}{\sum_i w_i(t)} \cdot N. \tag{4.9}$$

Thus, the number of workers assigned to station k is a certain fraction of all available workers. This fraction is determined by the number of work items at station k compared to all the work in the office. Equations (4.8) and (4.9) can be combined into the single equation

$$n_k(t) = \frac{w_k(t) \cdot N}{\max\{N, \sum_i w_i\}}. \tag{4.10}$$

This shows more clearly that if all $w_k(t)$ are continuous functions of t then so is $n_k(t)$. This continuity is necessary so that μ_k, as given by (4.6), is continuous, and hence the differential equation system (4.1) is uniquely solvable.

4.5. *Waiting Time*

Without some information about the manner in which work is processed at the workstations, it is not possible to estimate how long an item must wait for processing. The waiting time will depend, for example, on the queueing discipline and perhaps on the way in which resources are reallocated among the workstations. The differential equations (4.1) describe the change in the number of items at each station, but say nothing about the arrival or departure of a particular work item.

In order to illustrate how additional information can be used to estimate waiting time, suppose that the workstations are staffed by workers according to the allocation method described in §4.4. Furthermore, assume that work is processed on a first come first served basis. Then, if (4.7) holds, a new item arriving at any workstation can be processed immediately by an unassigned worker. However, if (4.7) does not hold then all workers are busy and a new item arriving at a station must wait.

Now, with n_k workers at station k, and per-person processing times of p_k hours per item, it follows that n_k/p_k items are processed in one hour. Hence, in h hours, $h \cdot n_k/p_k$ items are processed. Thus, a backlog of w_k items will be reduced to $w_k - h \cdot n_k/p_k$ in h

hours, and the total backlog at all workstations will be

$$\sum_k (w_k - h \cdot n_i/p_k). \tag{4.11}$$

A worker will be available for a newly arrived item as soon as (4.11) is less than the total number of available workers, i.e., as soon as

$$\sum_k (w_k - h \cdot n_k/p_k) < N. \tag{4.12}$$

By solving this for h we find

$$h > \frac{\sum_k w_k - N}{\sum_k n_k/p_k}. \tag{4.13}$$

More precisely, if at time t, there are backlogs of $w_k(t)$, with worker assignments n_k, then the waiting time for a new item is

$$WT = \begin{cases} 0 & \text{if } \sum_k w_k(t) \leqslant N, \\ \dfrac{\sum w_k(t) - N}{\sum n_k/p_k} & \text{otherwise.} \end{cases} \tag{4.14}$$

This, of course, assumes that the values of n_k do not change during the time period t to $t + WT$. Note that in the case where there are no workers of the type needed for workstation k, that is if $N = 0$, then (4.14) gives WT equal to infinity, which is reasonable.

5. Implementation Details

The model is implemented by a program that:

- Reads the data files that describe the work flow, processing times, resources, and external inputs.
- Sets up the starting conditions.
- Solves the differential equations over specified time intervals, reallocating resources as desired.
- Prints out tables of results.

The most time-consuming part of this is the solving of the differential equations. The numerical solution of this system proceeds in a step-by-step fashion from the initial time. For good accuracy, these steps must often be very small; on the order of 0.0001 hours for example. Thus, to solve the equations over, say, an eight-hour time period takes many thousands of steps. Each of these steps requires one or more evaluations of the right-hand side of the system (4.1). Hence, this evaluation should be programmed as efficiently as possible.

Another point to be kept in mind is that, under certain conditions, the system of differential equations (4.1) is *stiff*, as defined, for example in [6]. To illustrate the phenomenon of stiffness, and how it applies to (4.1), consider a very simple flow graph consisting of only one workstation. With constant arrival of $\lambda = \dot{\Lambda}(t)$ items per hour, a processing time of p hours per item per person, and n workers assigned to the station, the differential equation is

$$\dot{w}(t) = \lambda - n/p, \qquad 0 \leqslant t \leqslant T, \tag{5.1}$$
$$w(0) = 0.$$

Now, if there are a lot of workers available for assignment to the workstation, then

$n(t) = w(t)$, as in (4.8). Hence, the equation (5.1) is

$$\dot{w}(t) = \lambda - w/p \tag{5.2}$$

whose solution is

$$w(t) = \lambda p - \lambda p e^{-(t/p)}, \qquad 0 \leqslant t \leqslant T. \tag{5.3}$$

If p is *very* small, then the exponential part of (5.3) dies out very quickly, and the solution is essentially the constant λp. Unfortunately, this is a difficult kind of solution to compute numerically. The problem is that the differential equation solver thinks that the solution is constant: $w(t) = \lambda p$, so it takes very large time steps to integrate the equation. However, the numerical methods become unstable when applied to (5.2) with large time steps, so the computed solution will become very inaccurate. The outcome is that the differential equation solver will first take large time-steps, then repeat the integration with smaller and smaller steps, until the computed solution stabilizes. Overall, a great many steps will be used to compute this nearly constant solution.

This same situation holds more generally. That is, in the system of equations (4.1), if some of the μ_i are very small, then certain solutions $w_i(t)$ decay rapidly to steady state. This rapid decay causes instabilities in the numerical process. There are special differential equation solvers, based on so-called "stiffly stable" methods, that are designed for such systems. Unfortunately, they are expensive to use and require a large amount of computer memory. A better remedy is simply to omit workstations that have very short processing times.

6. Comparison with Other Methods

The flow problems to which this method can be applied can also be analyzed by more standard queueing theory techniques, and by discrete event or continuous simulations.

Simulation languages, such as GPSS, SIMSCRIPT, Q-GERT, GASP, DYNAMO, etc., have special constructs for describing information flow such as is shown in Figure 1. Arrival rates, processing times, and interfaces have to be carefully and completely defined, just as for the technique described here. The arrivals and processing times may, however, be stochastic, so that the effects of random arrivals and/or processing times can be studied. In addition, discrete simulations retain, as the name implies, the discrete nature of item arrivals and worker assignments. However, the use of simulation languages require special software packages that are usually very large and often costly. The differential equations model, on the other hand, can be easily programmed in any scientific programming language, such as FORTRAN. There are a variety of easily obtainable, well-documented, and thoroughly tested routines for solving the differential equations.

Much of the queueing theory literature is directed toward analyzing a few servers working independently or in rather simple combinations. Networks of queues have been analyzed by setting up a system of differential equations where the dependent variables represent the probability of queue length equal to n at the ith server at time t. See, for example, W. Fan [4]. This method, however, requires several differential equations for each workstation, so that for large networks the size of the differential equation system is too large to be practical. A queueing theory approach that leads to a set of differential equations similar to (4.1) is given by Chang [3].

For simplicity, we describe the method in [3] only as applied to a single server. In this case, Chang's analysis assumes that arrivals occur at random by a Poisson process, with time varying rate $\lambda(t)$, and the service time is exponentially distributed with mean

rate μ. The average queue length L is shown to satisfy the differential equation

$$\dot{L} = \lambda - \mu u \tag{6.1}$$

where u is the probability that the queue is not empty.

Now, a workstation with per-person processing time $p > 0$ and n workers can be thought of as a single server with service rate n/p. With this interpretation, (5.1) describes the same quantity as does (6.1); namely, the queue length at a single server. Keep in mind, however, that in (5.1), n/p is a fixed service rate, whereas in (6.1), μ is the mean value of an exponentially distributed service rate. The interesting point here is that if the service rate n/p in (5.1) is denoted by μ, then when $u = 1$ in (6.1) these two equations are identical. That is, when applied to a single server, the fluid flow technique with μ representing a constant service rate gives the same results as Chang's method when μ is the mean service rate, provided that the queue length is long (i.e., not empty with probability 1).

The purpose of this comparison is to illustrate a (simple) situation where the results produced by the fluid flow method are directly related to results given by more standard queueing theory methods. This discussion is not meant to suggest that either method should be used to analyze single server queues. There are, however, more complex queueing network problems that can be fruitfully analyzed by both the fluid flow technique and Chang's queueing theory based approach. As is often the case, however, when two methods can be used to analyze the same problem, there are tradeoffs to be considered. The fluid flow method can incorporate complicated worker allocation schemes, but cannot take into account random arrivals or variations in processing times. On the other hand, queueing theory based methods, such as Chang's, may require additional information, such as the value of u in (6.1), and are not suitable for examining the effect of reallocating resources such as workers. The choice of which method to use in a particular situation may not be clear-cut. The important point is that the fluid flow technique is an additional tool that can be useful in analyzing networks of queues.[1]

[1] Development of this modeling technique was supported by the Social Security Administration's Office of Assessment and Office of Advanced Systems. The cooperation and encouragement of these offices is sincerely appreciated.

References

1. BAILEY, N. T. J., *The Elements of Stochastic Processes with Applications to the Natural Sciences*, Wiley, New York, 1964.
2. BENDER, E. A., *An Introduction to Mathematical Modeling*, Wiley-Intersciences, New York, 1978.
3. CHANG, S. S. L., "Simulation of Transient and Time Varying Conditions in Queueing Networks," *Proc. 7th Annual Pittsburgh Conference, Modeling and Simulation*, Vol. VIII, Part II, Instrument Society of America, Pittsburgh, 1977.
4. FAN, W., "Simulation of Queueing Networks with Time Varying Arrival Rates," *Trans. International Association for Mathematics and Computers in Simulation*, Vol. XVIII, No. 3, North-Holland, Amsterdam, 1976.
5. FORSYTHE G., MALCOLM, M. AND MOLER, C., *Computer Methods for Mathematical Computations*, Prentice-Hall, Englewood Cliffs, N.J., 1977.
6. GEAR, C. W., *Numerical Initial Value Problems in Ordinary Differential Equations*, Prentice-Hall, Englewood Cliffs, N.J., 1971.
7. SHAMPINE, L. F. AND GORDON, M. K., *Computer Solution of Ordinary Differential Equations*, W. H. Freeman, San Francisco, 1975.
8. VANDERGRAFT, J. S., "An Analytic Model of SSA Service Delivery Units: Mathematical and Algorithm Details," ASG-TR-80-09, 1980.
9. ———, MIDDLETON, P. AND PANGALOS, S., "Staffing Analysis of SSA Service Delivery Units," ASG-TR-80-29, 1980.