Queuing theory and statistical methods are used in this paper to derive formulas for determining average turnaround time in teleprocessing systems that handle messages on a priority basis. This information is needed to ensure, for example, that messages are processed within acceptable time limits and that efficient use is made of system resources.

Factors considered in arriving at the formulas described here include waiting time in message queues, message processing time, I/O waiting time, and delays for higher priority processing. Results conform closely with those obtained from simulation studies.

Turnaround time for messages of differing priorities

by C. Hauth

Average turnaround time for processing messages of different priorities has far-reaching implications in the design of a teleprocessing system. Low average turnaround time for some types of messages may be highly desirable or even critical. For example, in an insurance application, the processing of claims, endorsements, and cancellations might each be done at different priorities. Moreover, a claims analyst might want claims to be processed within five minutes, so that he could assure an insurance agent that a claimant is covered. But thought must also be given to the kind of processing required by each priority class of message. If the kind of processing is not considered, system throughput may suffer. For example, if the processing required for higher priority messages involves little I/O activity and the messages are received at a sufficiently high rate, lower priority messages may never be processed.

Thus, many of the basic decisions that must be made in designing a useful and efficient teleprocessing system require a means of determining average turnaround time for processing messages of each priority in a given installation. The purpose of this paper is to assist in determining *average message turnaround time*—the interval between the time that a message is received and the completion of its processing in the computer.

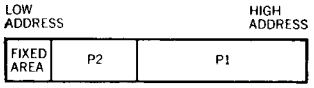
operating environment

The system assumed permits up to four separate jobs to be run concurrently within a single computing system having only

one central processing unit (CPU).[1] The separately scheduled jobs reside within their own predefined partitions in main storage, and the work done within each partition is assigned a different priority. While one job awaits completion of an event, such as an I/O operation, processing is switched to a lower priority job to take advantage of the temporary processing delay.

Figure 1 illustrates the main storage layout of a two-partition system. The higher-address partition is P_1, and the lower-address partition is P_2. Moreover, the job in partition P_1 has preferential access to the CPU; the job in partition 2 is given control of the CPU only when either there is no work to be done in the higher-priority partition or when the higher-priority job is suspended (in the wait condition). By dividing main storage into as many as four partitions, four separate jobs can be executed concurrently, with P_4 having the lowest priority.

Figure 1 Storage layout of a two-partition system



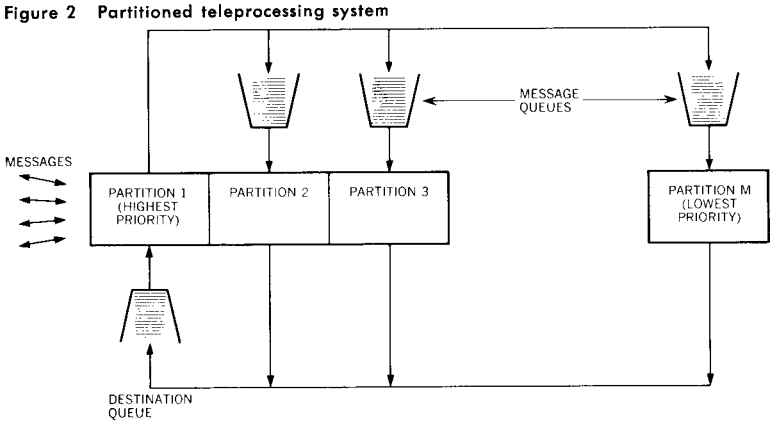
For teleprocessing, the highest priority partition, P_1, is normally occupied by a telecommunications control program,[2] which handles both the receiving and sending of messages. The remaining three partitions are occupied by message processing programs. For example, in the insurance application cited, one of the three partitions might handle claims, another endorsements, and the third cancellations.

In the system to be analyzed, all messages are received in partition 1 (the telecommunications control program). Then, messages with priority 1 are processed in partition 2, and messages with priority 2 are processed in partition 3. In general, messages with priority j are received in partition 1, processed in partition $j + 1$, and sent from partition 1. (An initial wait time precedes entry into any partition.) In the interest of generality, m partitions are considered here, rather than four. Turnaround time is developed for each priority class of messages. Note that the higher the number of the partition, the lower the priority.

Arrival rates to the partitions are assumed to have a Poisson distribution. Service time in the first partition is assumed to have an exponential distribution. The basic assumption here is that the distribution of all departing output messages from the partitions is also Poisson. Service times in the m partitions are assumed to be mutually independent. The sum of the service time in a particular partition and the initial wait time in the message queue for that partition is called *queuing time*. (Initial wait time should not be confused with wait time for I/O operations after processing in the partition has started.) From Figures 2 and 3, it is clear that data processing turnaround time for a message with priority j is equal to the sum of the receiving and sending queuing times for partition 1 and the queuing time for partition $j + 1$. Therefore, determining turnaround time for a message that will be processed in a particular partition is reduced to finding the queuing times for partition 1 and for that partition.

Note that for a system with only one partition, the analysis is essentially simple. The service time is the sum of the execution

Figure 2 Partitioned teleprocessing system

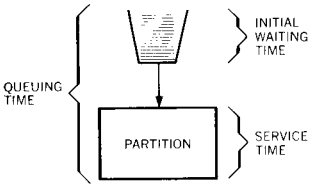


times for the problem program instructions. The initial wait time, W, for both sending and receiving can be found from the Pollaczek-Khintchine equation[3]

$$W = \frac{\lambda b_2}{2(1 - \lambda b_1)}$$

where λ is the arrival rate of messages to the partition, b_2 is the second moment of the service time for the partition, and b_1 is the mean of the service time for the partition. However, in a system with more than one partition, finding the queuing time for partition j, where $j > 1$, is much more complicated. The difficulty arises because both the initial waiting time and the service time for this partition may be extended due to interruptions originating in higher priority partitions. For example, if execution time for a macroinstruction in partition j is 5 milliseconds, it may be extended to 50 milliseconds due to interruptions from higher priority partitions.

Figure 3 Queuing time



It is important to note that if each partition contained no WAIT macroinstructions, the simple Preemptive-Resume Priority queuing equation could be used.[4]

$$T_{qj} = \frac{1}{1 - \sum_{i=1}^{j-1} \lambda_i b_{1i}} \left[b_{1j} + \frac{\sum_{i=1}^{j} \lambda_i b_{2i}}{2\left(1 - \sum_{i=1}^{j} \lambda_i b_{1i}\right)} \right]$$

where:

T_{qj} is the queuing time for partition j,

λ_i is the arrival rate to partition i,
b_{1i} is the mean of the service time for partition i,
b_{2i} is the second moment of the service time for partition i.

We first discuss the process of finding the service time for a partition (assuming that waiting will occur) and then that of finding the initial waiting time for each partition.

Service time

In general, the service time for a partition consists of processing time, wait time (nonoverlapped I/O time), and extensions of the service time due to interruptions from higher priority partitions. For example, execution of the following sequence of macroinstructions[5] might take place in a partition:

1. PROCESS (10 milliseconds)
2. GET TAPE FILE A (50 milliseconds)
3. PROCESS (20 milliseconds)
4. READ TAPE FILE B (60 milliseconds)
5. PROCESS (35 milliseconds)
6. WAIT (for READ to complete)

where service time in milliseconds is

$$10 + E_1 + 50 + E_2 + 20 + E_3 + 35 + E_5 + 25 + E_6$$

The time for the WAIT macroinstruction equals the READ time (60 milliseconds) minus the intervening processing time (35 milliseconds) or 25 milliseconds. E_i represents the extensions due to interruptions from higher priority partitions. Although determining processing time presents no particular difficulty, wait time and extensions to service time present a total of five problems:

1. Determining the extension of time for executing an input/output instruction due to requests for the same channel and/or device from other partitions.
2. Determining the WAIT macroinstruction time.
3. Determining the extension of time for a WAIT macroinstruction.
4. Determining the extension of time for a GET/PUT macroinstruction. (A GET/PUT has an embedded wait.)
5. Determining the extension of time for a PROCESS macroinstruction.

I/O time extensions

The extension of time for an input/output instruction in the partition under consideration due to input/output instructions in other partitions requesting the same channel and/or device is the delay caused by the channel and/or device being busy. The effect of I/O requests in higher priority partitions, lower priority partitions, and the same partition must be considered.

Each I/O instruction must be examined separately to determine its effect. The sum of the delays plus the actual I/O time constitutes the total time for this request. The influence of READ, WRITE, or EXCP macroinstructions (involving the same channel and/or device) in higher priority partitions can be seen from the example in Table 1.

Consider first the READ, WRITE, or EXCP requests in partition i (where $i = 1, 2, \cdots, j - 1$) that refer to the same channel or device. First, the difference between the I/O time and the intervening processing time (occurring before the corresponding WAIT) must be obtained. (If this difference is not positive, the request has no effect on READ_c.) Call this difference D_{ki}. If the request

Table 1 I/O activity in more than one partition

Partition i ($i < j$)	*Partition j*
Process_1 (1st PROCESS)	Process_1 (1st PROCESS)
.	.
.	.
.	.
READ_{ki} (kth I/O in partition i)	READ_c (I/O request being considered)
Process (intervening processing time)	.
WAIT_{ki}	.
.	.
.	.
.	.

(READ_{ki}) requires the same channel and device as the request in partition j (READ_c), the total added time is

$$0.5 \quad (\lambda_i s_i) D_{ki}(W_{ki}/T_i)$$

where λ_i is the arrival rate for partition i, s_i is the service time for partition i, W_{ki} is the wait time for the corresponding WAIT, and T_i is the total wait time for partition i (sum of the times for all WAIT macroinstructions). If the request does not require the same device, several conditions may be present. If $0.5D_{ki}$ is less than or equal to the data transfer time of the request in partition i, the added time is again

$$0.5\ (\lambda_i s_i) D_{ki}(W_{ki}/T_i)$$

If the remaining seek time for the I/O request in partition i is greater than the seek time for the request in partition j, the added time is zero. In all other cases, the additional time is equal to the data transfer time plus the remaining seek time for the I/O request in partition i minus the seek time of the I/O request in partition j multiplied by $\lambda_i s_i(W_{ki}/T_i)$. The added delay for the remaining types of input/output requests can be determined in a similar manner. The resulting formulas are tabulated in the Appendix.

wait time

The second problem encountered in determining the service time is determining the wait time. Consider the following example:

READ	TAPE FILE A (40 milliseconds)
PROCESS	25 milliseconds
WAIT	TAPE FILE A

In the above example, the WAIT time is the difference between the READ time and the intervening processing time or 15 milliseconds. The difficulty in calculating the WAIT time occurs when the associated I/O instruction has a nonzero variance. For example, an I/O request for a disk file normally has a nonzero variance. In this case, the mean of the maximum of the I/O time

and the intervening processing time must be obtained. From this, the intervening processing time is subtracted to determine the WAIT time. Since the I/O operation and the processing are occurring simultaneously, the WAIT time is equal to the time required for the longer of the two minus the intervening processing time.

In order to determine the mean of the maximum of the I/O time and the intervening processing time, the type of distribution of the I/O time must first be determined. This can be done by analyzing the variance and mean for the I/O time, and using one of six equations shown later. If the variance is zero, the distribution is constant and Equation 1 should be used. If mean^2/variance is greater than 5, the distribution is considered constant and again Equation 1 should be used. If $0 \leq$ mean^2/variance < 1.5, the distribution is considered exponential and Equation 2 should be used. If $1.5 \leq$ mean^2/variance < 2.5, the distribution is Erlang-2 and Equation 3 should be used. If $2.5 \leq$ mean^2/variance < 3.5, the distribution is Erlang-3 and Equation 4 should be used. If $3.5 \leq$ mean^2/variance < 4.5, the distribution is Erlang-4 and Equation 5 should be used. If $4.5 \leq$ mean^2/variance < 5, the distribution is Erlang-5 and Equation 6 should be used.

If the intervening processing time is represented as a and the mean of the I/O time as b, the first two formulas are:

$$\begin{aligned} &\text{mean of max} = \max(a, b) \\ &\text{variance of max} = 0 \end{aligned} \tag{1}$$

$$\begin{aligned} &\text{mean of max} = a + be^{-a/b} \\ &\text{variance of max} = b^2(2e^{-a/b} - e^{-2a/b}) \end{aligned} \tag{2}$$

In Equations 3 through 6, the distribution of the I/O time is Erlang-2 through Erlang-5, respectively. If the Erlang parameter is m, the general formula is:

$$\text{mean of max} = \int_0^\infty t dH(t)$$

where $H(t)$ is the distribution function of the maximum of $F(t)$ and $G(t)$, which are the distribution functions of the processing time and the I/O time, respectively. Note that the processing time and the I/O time are independent of each other. The distribution function of the maximum of $F(t)G(t)$, where $F(t)$ and $G(t)$ are independent, is equal to their product. Thus

$$H(t) = F(t)G(t)$$

$$F(t) = \begin{cases} 0 \text{ for } t < a \\ 1 \text{ for } t \geq a \end{cases}$$

$$G(t) = 1 - e^{-mt/b} \sum_{k=0}^{m-1} \frac{(mt/b)^k}{k!}$$

$$\therefore \int_0^\infty t dH(t) = \frac{m^m}{(m-1)!b^m} \int_a^\infty t^m e^{-mt/b} dt + a\left[1 - e^{-ma/b} \sum_{k=0}^{m-1} \frac{(ma/b)^k}{k!}\right]$$

Equations 3 through 6 represent the value of this expression for $m = 2, 3, 4, 5$, respectively.

$$\frac{4}{b^2}\left[\frac{ba^2e^{-2a/b}}{2} + \frac{b^2ae^{-2a/b}}{2} + \frac{b^3e^{-2a/b}}{4}\right] + a - ae^{-2a/b}\left(1 + \frac{2a}{b}\right) \tag{3}$$

$$\frac{3^3}{2b^3}\left[\frac{ba^3e^{-3a/b}}{3} + \frac{b^2a^2e^{-3a/b}}{3} + \frac{2b^3ae^{-3a/b}}{3^2} + \frac{2b^4e^{-3a/b}}{3^3}\right] + a - ae^{-3a/b}\left(1 + \frac{3a}{b} + \frac{9a^2}{2b^2}\right) \tag{4}$$

$$\frac{4^4}{3!b^4}\left[\frac{ba^4e^{-4a/b}}{4} + \frac{b^2a^3e^{-4a/b}}{4} + \frac{3b^3a^2e^{-4a/b}}{4^2} + \frac{6b^4ae^{-4a/b}}{4^3} + \frac{6b^5e^{-4a/b}}{4^4}\right] + a - ae^{-4a/b}\left(1 + \frac{4a}{b} + \frac{8a^2}{b^2} + \frac{32a^3}{3b^3}\right) \tag{5}$$

$$\frac{5^5}{4!b^5}\left[\frac{ba^5e^{-5a/b}}{5} + \frac{b^2a^4e^{-5a/b}}{5} + \frac{4b^3a^3e^{-5a/b}}{5^2} + \frac{12b^4a^2e^{-5a/b}}{5^3} + \frac{24b^5ae^{-5a/b}}{5^4} + \frac{24b^6e^{-5a/b}}{5^5}\right] + a - ae^{-5a/b}\left(1 + \frac{5a}{b} + \frac{25a^2}{2b^2} + \frac{125a^3}{6b^3} + \frac{625a^4}{24b^4}\right) \tag{6}$$

In Equations 3 through 6, the expressions enclosed in brackets represent the value of the integral

$$\int_a^\infty t^m e^{-m\,t/b}$$

The expressions enclosed in parentheses represent the value of the summation in the Erlang-m distribution evaluated at $t = a$, i.e.,

$$\sum_{k=1}^{m-1} \frac{1}{k!}\left(\frac{mt}{b}\right)^k$$

The general equation for the second moment of the maximum of a and b, where b has an Erlang-m distribution, is:

$$\frac{m^m}{(m-1)!b^m}\left[\frac{ba^{m+1}e^{-ma/b}}{m} + \frac{(m+1)b}{m}\int_a^\infty t^m e^{-m\,t/b}dt\right] + a^2 - a^2e^{-ma/b}\left[\sum_{k=1}^{m-1}\frac{1}{k!}\left(\frac{mt}{b}\right)^k\right]$$

The values for the integral and summation in the Erlang-m distribution obtained when calculating the mean can be substituted in this formula in order to obtain the second moment. The variance is then obtained by subtracting the square of the mean from the second moment.

The intervening processing time should then be subtracted from the mean of the maximum (a, b) to form the wait time for this I/O operation. The largest positive wait time of all the I/O operations waited for by this WAIT macroinstruction and the associated variance then represent the mean and variance of wait time for the WAIT macroinstruction. If there are no positive wait times corresponding to the I/O operations, the wait time for this WAIT macroinstruction is zero.

In this way, the wait time for each WAIT macroinstruction in partition j can be obtained. Thus, the second problem has been solved.

extensions to wait time

The third problem deals with finding the extension of time for a WAIT macroinstruction due to interruptions from higher priority partitions. If the WAIT macroinstruction is in partition 1, this extension is zero. However, if the WAIT macroinstruction is in a lower-priority partition, the wait time may be extended by interruptions from higher priority partitions. For example, if a WAIT macroinstruction takes 40 milliseconds but is interrupted for 50 milliseconds beginning after the first 10 milliseconds, the extension of the wait time is 20 milliseconds. Note that the extension is not 50 milliseconds, because use of the central processor by a higher priority partition while a lower priority partition is waiting has no effect on the lower priority partition.

Consider an interval of CPU time, z. Let b_i be the total CPU processing time (excluding I/O time and wait time) for partition i. Let λ_i be the arrival rate to partition i (assumed to have a Poisson distribution). Therefore, during an interval of CPU time, z milliseconds, partition i requires on the average $\lambda_i b_i z$ milliseconds. Therefore, partition i used the CPU a fraction of time equal to

$$\lambda_i b_i z/z \quad \text{or} \quad \lambda_i b_i$$

(Note that $\lambda_i b_i$ must be less than 1, or the system is over-utilized and lower-priority processing will not be done). Therefore, during an interval of CPU time, z milliseconds, partitions 1 through $j-1$ will use on the average

$$\sum_{i=1}^{j-1} \lambda_i b_i z \quad \text{milliseconds},$$

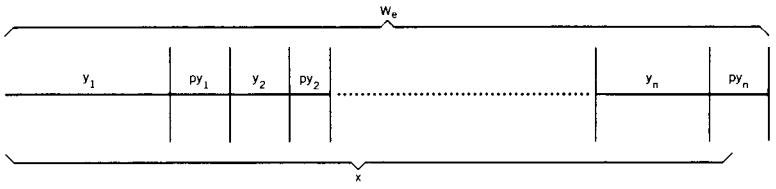
leaving

$$\left(z - \sum_{i=1}^{j-1} \lambda_i b_i z\right) \quad \text{milliseconds}$$

for partition j. Therefore, for every y milliseconds required by partition j,

$$\left[\frac{\sum_{i=1}^{j-1} \lambda_i b_i}{1 - \sum_{i=1}^{j-1} \lambda_i b_i}\right] y \quad \text{milliseconds}$$

Figure 4 Extended wait time



will be required by partitions 1 through $j - 1$. Let

$$p = \left\lceil \frac{\sum_{i=1}^{j-1} \lambda_i b_i}{1 - \sum_{i=1}^{j-1} \lambda_i b_i} \right\rceil$$

Therefore, p represents the fraction (which could be greater than 1) of CPU time taken from partition j by higher priority partitions. Note that for the first partition $p = 0$, and there is no extension of the wait time.

Let x be the value of the unextended wait time just calculated. The extended wait time can be thought of as broken up into portions y_j (as shown in Figure 4), where the computer is idle, and portions py_j, where the computer is being used by higher priority partitions. As an illustration, during the first 10 milliseconds of a 40-millisecond WAIT, the computer is idle; then, an interruption of 5 milliseconds occurs, followed by a 20-millisecond idle period and another interruption of 10 milliseconds. The extended wait time is therefore 45 milliseconds.

Thus, the extended wait time, W_e, can be represented as a sum of idle periods and interruptions as follows:

$$W_e = \sum_{j=1}^{n} \left(y_j + py_j \right)$$

where

$$\sum_{j=1}^{n} (y_j + py_j) \geq x \quad \text{and} \quad \sum_{j=1}^{n-1} (y_j + py_j) < x$$

However, it is only the last interruption that really affects the wait time. The interruptions that are overlapped with wait time have no effect on the time of the WAIT macroinstruction. For example, if the CPU were used by another partition for 5 milliseconds after 10 milliseconds of a 40-millisecond wait had passed, the WAIT macroinstruction would not be affected. However, if this interruption occurred after 37 milliseconds of the wait had passed, the wait time would be increased by 2 milliseconds.

Therefore, the extended wait time can be thought of as the sum of the last interruption time and the wait time occurring prior to the last interruption. If the wait is completed during an idle period, the last interruption is denoted as zero. Therefore, all

possibilities for the extended wait time can be represented as follows:

$$y + pz$$

where

$$\frac{x}{1+p} \leq y \leq x \quad \text{and} \quad \frac{x-y}{p} \leq z \leq y$$

In this equation, pz represents the last interruption and y represents the wait time prior to the last interruption. Since $y + pz \geq x$, y must be greater than or equal to $x/(1 + p)$. But y cannot be greater than x, because the wait would be ended at that point. Time z is actually the last idle period occurring before the final interruption, which is of length pz. Since $y + pz \geq x$, z must be greater than or equal to $(x - y)/p$. Since y is the wait time prior to the last interruption, z cannot be greater than y.

The average value of the function $y + pz$ over the region

$$[x/(1 + p) \leq y \leq x\,;(x - y)/p \leq z \leq y]$$

can be determined as follows. (Note that it is the average value of the extended wait time that is being represented by this function.)

$$\text{average value} = \frac{1}{A}\int_{x/(1+p)}^{x}\int_{(x-y)/p}^{y}(y + pz)dz\,dy$$

where

$$A = \int_{x/(1+p)}^{x}\int_{(x-y)/p}^{y} dz\,dy = \frac{x^2p}{2(1+p)}$$

$$\text{average value} = \frac{x}{3(1+p)^2}[p^3 + 5p^2 + 7p + 3]$$

Thus the average value of the extended wait time can be obtained. The mean of the extension is the average value minus x. The moment of extension of wait time is

$$\frac{1}{A}\int_{x/(1+p)}^{x}\int_{(x-y)/p}^{y}(y + pz - x)^2dz\,dy$$

Thus, the second moment is

$$\left[\frac{x^2(p^5 + 3p^4 - 3p^3 - 17p^2 - 18p - 6)}{6(1+p)^3}\right] + x^2$$

The variance of the extension is equal to the second moment minus the mean squared.

The average value of the extended wait time can be determined, and the variance of the extension of the wait time can be obtained. This variance can be added to the variance of the unextended wait time to give the total variance for this wait.

extensions to GET/PUT time

A GET/PUT macroinstruction has an imbedded WAIT macroinstruction. That is, no intervening processing is allowed between the start of the input/output operation and the WAIT. Therefore,

the input/output time of this macroinstruction is the same as the wait time for a WAIT macroinstruction. The computer is idle, waiting for the input/output operation to be completed. Therefore, the extension of this time due to interruptions from higher priority partitions can be determined in exactly the same way as the extension of the wait time.

extensions to PROCESS time

The last problem encountered in finding the service time for partition j is finding the extension of the time for a PROCESS macroinstruction due to interruptions from higher priority partitions. For example, if the computer is interrupted for a total of 5 milliseconds during execution of a PROCESS macroinstruction requiring 10 milliseconds, the effective time for the PROCESS macroinstruction is 15 milliseconds and the extension is 5 milliseconds. Note that for partition 1, a PROCESS macroinstruction cannot be extended, since there are no higher priority partitions. Let b_i be the mean of the total processing time for partition i (b_i excludes nonoverlapped I/O time and wait time). Let λ_i be the arrival rate (assumed to have Poisson distribution) at partition i. Therefore, CPU utilization for partition i is $\lambda_i b_i$. That is, during a time span of x milliseconds, partition i will use the CPU an average of $\lambda_i b_i x$ milliseconds. Therefore, the fraction of CPU time used by partition i is $\lambda_i b_i x/x$ or $\lambda_i b_i$. Therefore, the fraction of CPU time used by partitions 1 through $j - 1$ equals

$$\sum_{i=1}^{j-1} \lambda_i b_i$$

and partition j has the CPU available to it only a fraction of the time equaling

$$1 - \sum_{i=1}^{j-1} \lambda_i b_i$$

Assume that a PROCESS macroinstruction in partition j requires b_{jk} milliseconds. The extended processing time, b_{jk}^*, is equal to the sum of b_{jk} plus the average total interruption time from higher priority partitions. (The asterisk indicates that the time extension has been included.) But, during a time span of b_{jk}^* milliseconds, partitions 1 through $j - 1$ use the CPU for an average of

$$\sum_{i=1}^{j-1} \lambda_i b_i b_{jk}^* \quad \text{milliseconds}$$

$$\therefore b_{jk}^* = b_{jk} + \sum_{i=1}^{j-1} \lambda_i b_i b_{jk}^* = \frac{b_{jk}}{1 - \sum_{i=1}^{j-1} \lambda_i b_i} \quad \text{milliseconds} .$$

Note that for partition 1, $b_{jk}^* = b_{jk}$, since there are no higher priority partitions. The extension of time for a PROCESS macroinstruction in partition j requiring b_{jk} milliseconds can be determined as $b_{jk}^* - b_{jk}$, where b_{jk}^* is given above.

So far we have shown how to find the extensions of the time needed for input/output, PROCESS, and WAIT macroinstructions. The service time, s_j, for partition j is equal to the sum of the times for each macroinstruction plus the extension of time for each macroinstruction. The variance, v_j, is equal to the sum of the variances for each WAIT macroinstruction. Consider a partition containing the following instructions:

1	PROCESS	(30 milliseconds)
2	READ	TAPE FILE A (40 milliseconds)
3	PROCESS	(15 milliseconds)
4	WAIT	(25 milliseconds)

The mean service time is equal to the sum of 30 milliseconds, plus 15 milliseconds, plus 25 milliseconds, plus the extension of PROCESS #1, and the extension of WAIT. The variance of the service time, v_j, is equal to the variance of the wait time plus the variance of the extension of the time wait. (PROCESS times are assumed to be constant.)

Initial wait time

Since the service time for each partition can be found, the remaining problem is to determine the initial waiting time for each partition. The initial waiting time for partition j is the time a message must wait before being processed in this partition. After determining the service time and the initial waiting time, the queuing time can be determined as their sum.

Note that the initial waiting time for the first partition can be found by the Pollaczek-Khintchine equation:

$$w_1 = \frac{\lambda_1(s_1^2 + v_1)}{2(1 - \lambda_1 s_1)}$$

The size of the waiting line is $\lambda_1 w_1$. It will soon be shown that the initial waiting time for partition j is dependent on the initial waiting times for partitions 1 through $j - 1$. Therefore, the initial waiting time must be solved for partitions 1 through $j - 1$ before it can be solved for partition j. Since the value of the initial waiting time for partition 1 can be found as shown above, this presents no problem.

In addition to the service time, s_j, a message arriving at partition j and finding it free encounters an additional delay, W_{j1}, caused by the CPU being used by the higher priority partitions 1 through $j - 1$. For example, if a message arrives at partition 2 and no other messages are in the queue, but the CPU is currently processing in partition 1, the initiation of service for this message is delayed until processing in partition 1 is suspended. Note that the delay for partition 1 is zero, since there are no higher priority partitions.

We first develop the mean and second moment of this delay. Then we use the moments of this delay, the moments of the

Table 2 Illustration of process blocks

Partition		
PROCESS READ	10 msec FILE A	PROCESS BLOCK #1 = 10 msec
WAIT		
READ PROCESS	FILE B 30 msec	PROCESS BLOCK #2 = 30 msec
WAIT		
PROCESS	20 msec	PROCESS BLOCK #3 = 20 msec

service time, and the arrival rate to determine the initial waiting time.

In order to determine the moments of W_{j1}, it is necessary to examine the processing in the higher priority partitions. Any partition can be broken up into processing blocks. A processing block is the total CPU processing time and GET/PUT time that occurs: between 2 WAIT macroinstructions, prior to the first WAIT macroinstruction in the partition, or after the last WAIT macroinstruction in the partition, as shown in Table 2.

Only processing blocks affect the delay in the lower priority partition. If the higher priority partition is in a wait state, there is no delay to processing in the lower priority partition.

In partition i, call the first process block $i1$ and the kth process block ik. Let the number of process blocks in partition i be n_i. Let p_{ik}^* represent the time for processing block ik, which is extended to reflect interruptions from higher priority partitions.

Let λ_i be the arrival rate to partition i (where $i = 1, 2, \cdots, j - 1$). When a new message arrives at partition j and there are no prior messages being served or in the queue for this partition, a delay may be encountered because a process block in a higher priority partition is being executed. If all higher priority partitions are in a wait condition or are not busy, they do not affect the new arrival. The probability that partition i (where $i = 1, 2, \cdots, j - 1$) is handling a message is $\lambda_i s_i$, where s_i represents the service time for partition i. Since processing block ik represents a fraction of the service time equal to p_{ik}^*, the probability that processing block ik is being executed is

$$\lambda_i s_i(p_{ik}^*/s_i) = \lambda_i p_{ik}^*$$

Assuming that, on the average, half of the processing block remains to be processed, the average total delay, D_{ik}, caused by the CPU being busy processing this block is $0.5p_{ik}^*$.

If the CPU is busy executing the last process block of partition i, a further delay d_{j2} may result if there is another message waiting to be served in partition i. The CPU executes the first process

Table 3 Partition states

State	*Probability* (P_{ik})	*Delay* (D_{ik})
$e_{ik}(1 \leq k \leq (n_i - 1)$ Block k is being processed in partition i.	$\lambda_i p_{ik}{}^*$	$0.5p_{ik}{}^*$
e_{in_i} Last block is being processed in partition i, and nothing is in the waiting line for this partition.	$\lambda_i p_{in_i}{}^* e^{-\lambda_i(s_i+w_i)}$	$0.5p_{in_i}{}^*$
$e_{i,(n_i+1)}$ Last block is being processed in partition i, and there are one or more messages in the waiting line for this partition.	$\lambda_i p_{in_i}{}^*[1 - e^{-\lambda_i(s_i+w_i)}]$	$0.5p_{in_i}{}^* + p_{i1}{}^*$
$e_{i,(n_i+2)}$ Probability that partition i is free with a message(s) in the waiting line.	$(1 - \lambda_i s_i)\lambda_i w_i$ for $i > 1$ 0 for $i = 1$	$p_{i1}{}^*$

block $i1$ in the partition for the new message after it has executed the last block for the previous message. The probability that the last block in partition i, (in_i), is being executed is $\lambda_i b_{in_i}{}^*$. The probability of at least one arrival for this partition during the initial waiting time and service time of the prior message is

$$1 - e^{-\lambda_i(w_i+s_i)}$$

where w_i represents the initial wait time for partition i and s_i represents the service time for partition i. (With a Poisson arrival rate, λ_i, the probability of no new arrivals during time t is $e^{-\lambda_i t}$.) Therefore, the delay, d_{j2}, caused when the CPU is busy executing the last process block in partition i and when there is another message waiting to be served by partition i, is equal to $p_{i1}{}^*$.

If the CPU is busy executing a process block for partition h, (where $h = 1, 2, \cdots, j - 2$), and partition $h + 1$ is free with a new arrival waiting to be served, an additional delay results for the newly arriving message for partition j. Processing in partition j cannot be initiated until processing in partition $h + 1$ has been initiated and suspended. The probability that partition $h + 1$ is free is $1 - \lambda_{h+1}s_{h+1}$, since the utilization for this partition is $\lambda_{h+1}s_{h+1}$, i.e., the product of the arrival rate and the service time. The probability that there is a message in the waiting line for this partition is $\lambda_{h+1}w_{h+1}$, where w_{h+1} represents the initial waiting time for this partition. Therefore, the probability of this occurrence is $(1 - \lambda_{h+1}s_{h+1})(\lambda_{h+1}w_{h+1})$, and the associated delay is $p_{h+1,1}{}^*$ (the length of the first processing block).

Therefore, when a message for partition j arrives and partition j is free, partition i (where $i = 1, 2, \cdots, j-1$) can be in any of $n_i + 2$ states, which results in a delay in the processing of the message for partition j. The states, e_{ik}, and the associated probability and delay associated with each are shown in Table 3.

Therefore, the mean of the delay (w_{j1}) encountered by a message arriving at partition j and finding it free is equal to the summation of all permutations and combinations of the probabilities, P_{ik}, multiplied by the sum of the associated delays. (Note that $i_1 \neq i_2$ and $i_f \neq i_g$ in the summations below.)

$$\therefore w_{j1} \text{ (mean of } w_{j1}) =$$

$$\sum_{i=1}^{j-1}\sum_{k=1}^{n_i+2} P_{ik}D_{ik} + \sum_{i_1=1}^{j-1}\sum_{i_2=1}^{j-1}\sum_{k_1=1}^{n_i+2}\sum_{k_2=1}^{n_i+2} P_{i_1k_1}P_{i_2k_2}(D_{i_1k_1} + D_{i_2k_2})$$

$$(i_1 \neq i_2)$$

$$+ \cdots + \sum_{i_1=1}^{j-1}\sum_{i_2=1}^{j-1}\cdots\sum_{i_t=1}^{j-1}\sum_{k_1=1}^{n_{i_1}+1}\sum_{k_2=1}^{n_{i_2}+1}\cdots\sum_{k_t=1}^{n_{i_t}+1} P_{i_1k_1}P_{i_2k_2}\cdots P_{i_tk_t}(D_{i_1k_1} + D_{i_2k_2} + \cdots + D_{i_tk_t})$$

$$(i_f \neq i_g \text{ for } f \neq g)$$

$$+ \cdots + \sum_{k_1=1}^{n_1+2}\sum_{k_2=1}^{n_2+2}\cdots\sum_{k_{(j-1)}=1}^{n_{(j-1)}+2} P_{1k_1}P_{2k_2}\cdots P_{(j-1)k_{(j-1)}}(D_{1k_1} + D_{2k_2} + \cdots + D_{(j-1)k_{(j-1)}})$$

$$\overline{W}_{i1}^2 \text{ (2nd moment of } W_{j1}) =$$

$$\sum_{i=1}^{j-1}\sum_{k=1}^{n_i+2} P_{ik}D_{ik}^{\,2} + \sum_{i=1}^{j-1}\sum_{i_2=1}^{j-1}\sum_{k_1=1}^{n_{i_1}+2}\sum_{k_2=1}^{n_{i_2}+2} P_{i_1k_1}P_{i_2k_2}(D_{i_1k_1} + D_{i_2k_2})^2$$

$$(i_1 \neq i_2)$$

$$+ \cdots + \sum_{i_1=1}^{j-1}\sum_{i_2=1}^{j-1}\cdots\sum_{i_t=1}^{j-1}\sum_{k_i=1}^{n_{i_1}+2}\cdots\sum_{k_t=1}^{n_{i_t}+2} P_{i_1k_1}P_{i_2k_2}\cdots P_{i_tk_t}(D_{i_1k_1} + D_{i_2k_2} + \cdots + D_{i_tk_t})^2$$

$$(i_f \neq i_g \text{ for } f \neq g)$$

$$+ \cdots \sum_{k_1=1}^{n_1+2}\sum_{k_2=1}^{n_2+2}\cdots\sum_{k_{(j-1)}=1}^{n_{(j-1)}+2} P_{1k_1}P_{2k_2}\cdots P_{(j-1)_1k_{(j-1)}}(D_{1k_1} + D_{2k_2} + \cdots + D_{(j-1)k_{(j-1)}})^2$$

At this point, the mean s_j and variance of the service time for partition j (where $j = 1, 2, \cdots, m$) have been found in addition to the mean w_{j1} and second moment $\overline{W}_{j1}^2$ of the delay of a message that arrives at partition j and finds it idle. Note that a message that arrives at partition j when it is busy must wait for all of the previous services for partition j to be completed. However, as soon as the preceding message has been processed, the newly arrived message is immediately processed. In this case, there is no possibility of interruptions from higher priority partitions. The interval between the termination of one message and the initiation of processing of the next message in a partition is considered to be zero for this analysis. Therefore, since the CPU belongs to partition j at the termination of processing of a message, partition j can immediately begin processing the next message in the queue.

In order to determine the initial waiting time, including delay (W_{j1}), some previously developed results have been used.[6]

Table 4 Comparison of analysis and simulation results

Test case	*Number of partitions*	*Service macroinstructions*	*Partition number*	*Arrival rates (msg/msec)*	*Queuing times (msec)*		*Largest percentage difference*
					analysis	*simulation*	
1	2	tape I/O	1	0.004	43.477	43.613	4
		WAIT	2	0.013333	46.307	44.484	
		PROCESS					
2	2	tape I/O	1	0.008	48.666	49.578	less than 1
		WAIT	2	0.006667	42.151	42.414	
		PROCESS					
3	2	disk I/O	1	0.001	79.898	80.713	4.9
		WAIT	2	0.001	92.005	96.751	
		PROCESS					
4	3	tape I/O	1	0.008	48.977	49.302	5
		WAIT	2	0.006667	43.021	41.729	
		PROCESS	3	0.006667	55.273	58.369	
5	3	tape I/O	1	0.004	43.717	43.372	4.8
		WAIT	2	0.013333	47.189	46.104	
		PROCESS	3	0.006667	62.889	66.070	
6	10	tape I/O to					
		3 channels	1	0.00096	40.22	40.13	5
		WAIT	2	0.00104	31.47	31.29	
		PROCESS	3	0.00098	32.39	32.52	
			4	0.00097	33.32	33.25	
			5	0.00102	34.33	34.16	
			6	0.00102	34.71	35.48	
			7	0.00087	35.73	36.73	
			8	0.00095	36.79	37.40	
			9	0.00095	37.94	39.45	
			10	0.00102	39.21	41.30	

The assumptions and results are as follows. Assume a single partition to service messages (partition j where $j = 1, 2, \cdots, m$); Poisson distributed input; and a queue with input parameter λ_j and with a service distribution function $G_j(x)$ (mean $= s_j$, variance $= v_j$). Suppose, in addition, that if a message arrives and finds the partition idle, the message has to wait before servicing of it begins a random delay with distribution function $W_j(x)$. Further, suppose that if a message arrives and the partition is busy, it must wait until the message ahead has been serviced and an additional random delay with a distribution function $H_b(x)$ before servicing of it begins. (For this system, the delay with distribution function $H_b(x)$ is a constant zero.) Let $\phi(s)$, $u_e(s)$, and $u_b(s)$ be the Laplace-Stieltjes transforms of $G_j(x)$, $W_j(x)$, and $H_b(x)$, respectively. Then, the Laplace-Stieltjes transform $\eta(s)$ of the initial waiting time (including the delay for service) is as follows:

$$\eta(s) = \frac{1 - \lambda_j s_j}{1 + \lambda_j W_{j1}} \left[\frac{\lambda_j [u_e(s) - u_b(s)] - s u_e(s)}{\lambda_j - s - \lambda_j \phi(s) u_b(s)} \right]$$

Since $H_b(x)$ is the distribution function for a constant 0, $u_b(s) = 1$. The mean of the initial waiting time is equal to $(-1)\eta'(s)$, $s = 0$.

$$\eta'(s) = \frac{1 - \lambda_j s_j}{1 + \lambda_j w_{j1}} \left[\frac{[\lambda_j - s - \lambda_j \phi(s)][\lambda_j u_e'(s) - s u_e'(s) - u_e(s)] - \{\lambda_j [u_e(s) - 1] - s u_e(s)\}[-1 - \lambda \phi'(s)]}{[\lambda_j - s - \lambda_j \phi(s)]^2} \right]$$

Using L'Hospital's rule twice:

$$(-1)\eta'(0) = \frac{[(1 - \lambda s_j)(\lambda_j \overline{W}_{j1}^2 + 2w_{j1}) + \lambda_j (1 + \lambda_j w_{j1})(s_j^2 + v_j)]}{2(1 - \lambda_j)(1 + \lambda_j w_{j1})}$$

The expression $(-1)\eta'(0)$ represents the mean of the initial waiting time, including the delay for service.

At this point, the mean of service time s_j has been found, and the initial waiting time has been found for partitions 1 through m. The queuing time for the partition is simply the sum of these two results.

Once the queuing time for each partition has been found, the message data processing turnaround time for a message with priority j can be found by summing the queuing times for partitions 1 and $j + 1$.

In order to validate the analysis, message queuing time was tested using the formulas and simulation. Identical models were used with the results in Table 4.

Summary

Performing data processing services at different priorities involves considerations not usually associated with queuing theory. Basically, this is due to the fact that a lower priority service can be performed at the same time that an input/output operation is taking place as part of a higher priority service. This overlapping of I/O operations with the execution of instructions complicates the problem of obtaining quantitative information about data processing services done on a priority basis. Yet this kind of information is highly desirable for designing efficient systems.

Our problem was to find the average time required to process a message of a given priority in a system that can process messages at different priorities (average turnaround time). We find average turnaround time by adding together all of the component times that elapse in processing a message of a given priority.

In the system under consideration, all messages are processed in both a control program partition and a processing partition of main storage. Total turnaround time here is the sum of the queuing times for these partitions. Queuing time, in turn, is the initial waiting time before processing in a partition begins plus the processing time after the message has been accepted into the partition.

Determining initial waiting time for the highest priority partition is relatively simple. However, finding initial waiting time for lower priority partitions is complicated by the fact that this time may be increased because of interruptions by higher priority processing.

Service time is the sum of the times required to execute instructions within the partition and the times spent waiting for I/O operations that are not overlapped with program execution. However, both of these times may also be extended because of interruptions from higher priority partitions.

The equations used here to determine each of these times take account of factors frequently encountered in performing data processing tasks on a priority basis. Thus, the approach used to solve this problem is applicable to many comparable problems. Moreover, many of the equations can be applied directly where identical conditions prevail.

CITED REFERENCES

1. *IBM SYSTEM/360 Operating System—Option 2: Multiprogramming with a Fixed Number of Tasks—Concepts and Considerations*, C27-6926-0, IBM Data Processing Division, White Plains, New York.
2. W. P. Margopoulos and R. J. Williams, "On teleprocessing system design," *IBM Systems Journal* **5**, No. 3, 134–141 (1966).
3. T. T. Saaty, *Elements of Queueing Theory*, McGraw-Hill Book Company, Inc., New York, New York (1961).
4. *IBM Data Processing Techniques—Analysis of Some Queuing Models in Real-Time Systems*, F20-0007, IBM Data Processing Division, White Plains, New York.
5. "The functional structure of OS/360."
   G. H. Mealy, "Part I, Introductory survey," IBM *Systems Journal* **5**, No. 1, 2–11 (1966).
   B. I. Witt, "Part II, Job and task management," *ibid.*, 12–29.
   W. A. Clark, "Part III, Data management," *ibid.*, 30–51.
6. P. Welch, "On a generalized queuing process in which the first customer of each busy period receives exceptional service," *Operations Research* **12**, No. 5, 736–752 (September-October 1964).

GENERAL REFERENCES

1. L. Takacs, *Introduction to the Theory of Queues*, Oxford University Press, New York, New York (1962).
2. W. Feller, *An Introduction to Probability Theory and Its Applications*, **1**, John Wiley & Sons, New York, New York (1960).

Appendix

Equations for the extension of time A for an input/output instruction caused by other requests for the same channel and/or device are stated below.

The following notation applies in the these equations:

j partition containing request for which added time is being calculated.

ki request that is influencing the request in partition j.

U_{ki} input/output time of request ki.

B_{ki} blocking factor for request ki.
W_{ki} time for WAIT macroinstruction associated with request ki.
T processing time occurring between the I/O macroinstruction in partition j and the prior WAIT macroinstruction.
R_{ki} data transfer time for request ki
T_p processing time occurring between I/O request kj and the I/O macroinstruction in partition j.
S_{kj} seek time for request kj.
I_{kj} index seek and read time for request kj.
S_{cj} seek time for request in partition j.

1. GET/PUT instructions on the same channel and device—higher priority partitions:

$$A = \left[1 + \frac{U_{ki}}{B_{ki}(1/\lambda_i + W_{ki})}\right] U_{ki} - (0.5 + L)B_{ki}(1/\lambda_i W_{ki})$$

where

$$L = \frac{U_{ki}}{B_{ki}(1/\lambda_i + W_{ki})},$$

which is evaluated first and the result is truncated.

2. GET/PUT instructions on the same channel—higher priority partitions:

$$A = U_{ki} - (0.5 + L)B_{ki}(1/\lambda_i + W_{ki})$$

where

$$L = \frac{U_{ki}}{B_{ki}(1/\lambda_i + W_{ki})},$$

which is evaluated first and the result is truncated.

3. READ/WRITE or EXCP instructions on the same channel and device—lower priority partitions.

$$A = \left(\frac{\lambda_i {U_{ki}}^2}{2}\right) - T$$

4. READ/WRITE or EXCP instructions on the same channel—lower priority partitions.

$$A = \left(\frac{\lambda_i U_{ki} R_{ki}}{2}\right) - T$$

5. GET/PUT instructions on the same channel and device—lower priority partitions:

$$A = (L - 1)\left(U_{ki} - \frac{B_{ki}}{\lambda_i}\right) + \left(\frac{0.5\lambda_i {U_{ki}}^2}{2}\right)$$

where

$$L = \frac{U_{ki}}{B_{ki}(1/\lambda_i)},$$

which is evaluated first and the result is truncated.

6. GET/PUT instructions on the same channel—lower priority partition:

$$A = U_{ki} - L\left(\frac{B_{ki}}{\lambda_i}\right)$$

where

$$L = \frac{U_{ki}}{B_{ki}(1/\lambda_i)},$$

which is evaluated first and the result is truncated.

7. READ/WRITE or EXCP instructions on the same channel and device—same partition (with no intervening wait).

$$A = U_{kj} - T_p$$

8. READ/WRITE or EXCP instructions on the same channel—same partition (with no intervening wait).

$$A = U_{kj} - T \qquad \text{for } I_{kj} + S_{kj} \leq T_p$$
$$A = U_{kj} - T_p - S_{cj} \qquad \text{for } I_{kj} > T_p,\ S_{kj} > S_{cj}$$
$$A = U_{kj} - T_p - S_{cj} \qquad \text{for } U_{kj} \leq T_p,\ U_{kj} - I_{kj} \leq T_p + S_{cj}$$

9. GET/PUT instruction on the same channel and/or device—same partition (occurring before the request being considered).
   A will be calculated as in 8 and divided by the blocking factor.
10. GET/PUT instruction on the same channel and/or device—same partition (occurring after the request being considered).
    A will be calculated as in 8 and decremented by $(1/\lambda_i - 2T_p)$. This result will be divided by the blocking factor to form the added time.