In typical teleprocessing applications, a large number of terminals communicate with the main-storage unit of a centrally located computer. The communication activities may reduce the processing capacity of the computer by claiming a significant number of main-storage access cycles.

If the number of cycles to be claimed is partially dependent upon channel-busy and request-pending conditions, the problem involves a probability analysis. Using a simple queuing model as a starting point, a usefully accurate solution method is developed. The SYSTEM/360 *multiplexor channel is analyzed as an example.*

On teleprocessing system design

Part V A technique for estimating channel interference

by T. W. Gay, Jr.

The function of data channels is to effect the transfer of data between input/output devices and main storage. In fulfilling this function, data channels ordinarily make use of main storage, the extent of use depending upon channel characteristics as well as data rates. Let u denote the proportion of main-storage cycles claimed for data-channel operations; for lack of a more descriptive term, u is most widely known as *channel interference.* In choosing a system configuration for the needs of a specific installation or application, it is often desirable to estimate u.

Let λ denote the average rate of data flow between the input/output devices and main storage of a system configuration. If the data channels are so designed that the number of main-storage accesses for data operations is a linear function of λ, the determination of u is trivial. If u is a linear function of λ_1, λ_2, $\cdots$, λ_i, $\cdots$, λ_N, where λ_i is the average data rate for the ith combination of device types and record formats, the solution is less elegant but no more difficult in principle. The later formulation is closely applicable to a SYSTEM/360 configuration with selector channels only.

A more difficult problem arises if u is a function of channel status as well as of λ. To cite an example, let ϕ and ψ denote status values: $\phi = 1$ for channel busy, $\phi = 0$ for channel not busy; $\psi = 1$ if a request is pending, $\psi = 0$ if no request is pending. In this case, u is given by some function $f(\lambda, 0, \psi)$ because the number of memory cycles required to serve a request on a

SYSTEM/360 multiplexor channel can depend upon ϕ and ψ as well as the number of data bytes involved in the request.

Using queuing theory, Chang[1] has given a model that permits a general solution to $f(\lambda, \phi, \psi)$. The purpose of this paper is to provide an approximate solution that is sufficiently accurate for system design purposes and has the additional advantages of a highly intuitive mathematical model. The multiplexor channel is described at the level of detail required for use as an example of the method.

A model for the multiplexor channel

In the *multiplex mode*, the multiplexor channel can concurrently handle data for many slow-speed devices; in the *burst mode*, it can serially handle data for one high-speed device. The burst mode can be analyzed in much the same fashion as a selector channel, but is seldom utilized in a telecommunications environment. The multiplex mode permits two optional choices of transfer operation: single-byte and multibyte.

In single-byte transfer, the channel accepts requests from concurrently operating devices and services the requests one at a time—each request involving a transfer of one byte. The single-byte case is particularly relevant to telecommunication applications and the one we intend to analyze at more length. The multibyte transfer mode, designed for buffered devices such as printers and card readers, may imply the transfer of more than one byte per request. Multibyte transfer may be analyzed using methods similar to the one to be given for single-byte transfer.

The length of time required for the multiplexor channel to service a byte is a function of channel status. If an arriving (incoming or outgoing) byte finds the channel idle, elapsed time t_1 is required to store computer registers. If the arriving byte finds the channel busy, $t_1 = 0$ because the registers have already been stored.

For each request, time interval t_2 elapses while the multiplexor channel fetches a unit control word (UCW), transfers the byte, updates the UCW, stores the updated UCW, and tests for a pending service request.

If the test for a pending service request is negative, an interval t_3 elapses while the channel restores the computer registers and ends multiplexor service. If the test is positive, the channel immediately begins servicing the pending request; no time is required to store computer registers because the registers are still available for use.

The schematic shown in Figure 1 illustrates a single-server queuing system with three stages of service, where entry to Stages 1 and 3 is dependent upon channel status. For the time being, we assume that selector-channel activity is zero. Knowing that a busy multiplexor channel preempts main storage, we may let v denote the probability of the multiplexor channel being busy as well as the value of channel interference. Then, assuming that

the probability of a pending request (Stage 3) is independent of the probability of a busy channel, we may write

$$v = \lambda[(1 - v)t_1 + t_2 + (1 - v)t_3]$$

Expanding this expression and solving for v, we have

$$v = \frac{\lambda(t_1 + t_2 + t_3)}{1 + \lambda(t_1 + t_3)} \tag{1}$$

Interpretation of Equation 1 shows that multiplexor channel utilization v is the product of the service request throughput λ and the equivalent mean service time per byte, t, where t is given by

$$t = \frac{(t_1 + t_2 + t_3)}{1 + \lambda(t_1 + t_3)}$$

The maximum of t occurs where $\lambda = 0$, at which point $t = t_1 + t_2 + t_3$. The minimum occurs where λ is large enough to make $v = 1$, at which point $t = t_2$.

It is worth noting of the model that, for given λ, slowing down the main storage would have the effect of increasing the probability of YES branches. As a result, doubling the time of each main-storage access (and related control functions as well) will less than double v. The same effect occurs if other devices with higher priorities than the multiplexor channel steal a proportion w of the memory cycles. From the vantage point of the multiplexor channel, this has the effect of stretching memory cycles by the factor $1/(1 - w)$.[2] If we divide each t in Equation 1 by $1 - w$ and simplify, we obtain

$$v^* = \frac{\lambda(t_1 + t_2 + t_3)}{(1 - w) + \lambda(t_1 + t_3)} \tag{2}$$

where v^* denotes interference in a hypothetical machine that is slower than the true machine. To put it another way, the multiplexor channel has available a fraction $(1 - w)$ of the true machine, and to obtain v we prorate the multiplexor interference of (2) over the true machine such that

$$v = v^*(1 - w) \tag{3}$$

For example, let t_1, t_2, and t_3 equal 15.00, 31.50, and 15.75 microseconds, respectively. Let λ equal 8000 bytes per second, and let w be 0.25. Then, by (2) and (3),

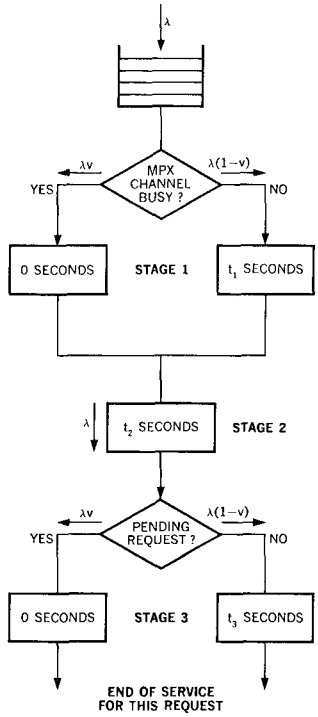
$$v^* = \frac{8(15.00 + 31.50 + 15.75)(10^3 \times 10^{-6})}{1 - 0.25 + 8(15.00 + 15.75)(10^3 \times 10^{-6})} = 0.50$$

$$v = 0.50(1 - 0.25) = 0.3725$$

Had w been zero, Equations 2 and 3 would yield $v = 0.40$, as would Equation 1.

The combined interference due to both w and v is simply $w + v$ or 0.6225. In SYSTEM/360, where selector channels have higher priorities than the multiplexor channel, the factor w could arise from selector channels.

Figure 1 Schematic timing model of multiplexor channel



The method given for estimating v assumes that the event of a busy multiplexor channel is independent, in the probability sense, of the event of a pending request. This would be a poor assumption if, for example, requests came in at periodic points in time. It is a realistic assumption only if requests are being generated haphazardly by a number of independent lines. Assuming a Poisson distribution of request arrivals, the results were found to agree within three percent with comparable results obtained from Chang's queuing model. The agreement becomes even better as v approaches either zero or unity.

After the average amount of channel interference has been estimated, it is still of interest to assess the "worst-case" conditions in which high-priority devices operating concurrently claim all of the memory cycles for a significant period of time. Such an analysis can usually be carried out by simply examining record lengths, device characteristics, and maximum request rates.

Summary

The mean percentage of main-storage cycles required by a multiplexor channel is dependent upon the status of the channel at service time as well as the rate of incoming requests and the interference caused by channels of higher priority. A simple and reasonably accurate method of estimating multiplexor channel interference is discussed. The method is based on a mathematical model which assumes that requests for multiplexor service arrive haphazardly from a number of requesting sources; this assumption is realistic for most teleprocessing applications.

ACKNOWLEDGMENT

The author wishes to thank W. Chang and P. H. Seaman for help in the form of technical discussion and guidance.

CITED REFERENCE AND FOOTNOTE

1. W. Chang and D. J. Wong, "Computer channel interference analysis," *IBM Systems Journal* 4, No. 2, 162–170 (1965).
2. If n memory cycles are required by a particular server in processing one transaction in one unit of time, and if $(1 - w)$ memory cycles are available per unit of time, than n units of time will be required to complete the transaction service. Because $x = n(1 - w)x$, we have $n = 1/(1 - w)$.