J. H. Chang

# Terminal Response Times in Data Communications Systems

**Abstract:** A response time analysis for a general class of terminals-to-computer subsystem is presented in this paper. The model used is based on the most advanced data communications system in which terminals are connected to Terminal Control Units (TCU) that are in turn connected to local Front-End Processors (FEP). The line control procedures used to interface a TCU and an FEP may be half-duplex Binary Synchronous Communications (BSC), half-duplex Synchronous Data Link Control (SDLC), or full-duplex SDLC. The models presented here can be used to determine bottlenecks in the entire system and to facilitate the initial phase of system design and configuration.

## Introduction

A generic configuration of data communications systems consists of many components such as terminals, Terminal Controller Units (TCU), communications lines, remote as well as local Front-End Processors (FEP), host processors, and auxiliary storage devices (see Fig. 1). Each one of these components has its own specifications and operating characteristics in terms of data rate, transmission media, and functional capabilities. One of the key factors in design and evaluation of such systems is the calculation of terminal response time, which can be defined as the time interval from the operator's pressing the last key (send key) of the input to the terminal's typing or displaying the first character of the response. Systems differ widely in their response time requirements, and the response time needed can, in turn, have a major effect on the design of the data transmission network and the data processing facilities. This paper presents the development of an analytical framework for analyzing response time requirements of data communications systems.

The terminal response time as defined above is the totality of several time elements. At the time when the send key is depressed, the complete transaction has already been stored at a prespecified buffer area in the TCU, one for each terminal. Transactions stored at their terminal buffers cannot be transmitted to the host site until the particular TCU at which these transactions reside is polled by the local FEP in accordance with a given polling list. The time spent by a transaction waiting for polling is the first time element to be calculated in obtaining the total terminal response time. This element depends on the system configuration and line procedures. The time required for transaction transmission along communications lines is relatively easy to calculate once we know the length of a transaction and the line speed. When a transaction arrives at an FEP, certain delays may occur because this is where most communications functions are performed. In case the FEP is a simple control unit whose only function is character assembly and disassembly, such delay would be negligible. After the whole transaction has entered the host processor, its processing time depends on the application programs, CPU processing speed, operating system, access methods, and the characteristics of the auxiliary storage devices such as disk files. A completely processed transaction will then wait at the FEP until the addressed line and TCU are ready to receive their responses. The length of this waiting time can generally be analyzed by an approximate queuing model.

Readers familiar with teleprocessing systems are aware of the fact that there are many different variations in system configurations and operations. The connection between the TCU and FEPs may be in the form of loops, stars, or multi-drops. The mode of transmission may be half-duplex or full-duplex with different line control procedures such as Binary Synchronous Communications (BSC) and Synchronous Data Link Controls (SDLC). After an inquiry has been sent to the host site for processing, the whole communication path may be held throughout the entire question-answering period, or the sending terminal may release the communication path, in whole or in part, so that other terminals and TCUs can send and receive their transactions. It is assumed in this paper that a terminal will release its line after its transaction is keyed in, and that several TCUs share the same high speed line in a multi-drop fashion to

communicate with the local FEPs. However, both the half-duplex and full-duplex modes can be accommodated by the analysis.

In a recent survey paper, Green and Tang [1] discussed in depth the state of the art in using analytical models to design terminal-oriented systems. They divided the whole design and configuration process into two parts: the network models and the host models. Various problem areas and the progress to date were summarized. It was indicated that there has been no overall treatment of complete computer communications systems that would allow one to carry out the configuration process, taking into account details of the various transactions within the system. To the best knowledge of the author, the present paper represents the first attempt to bridge the gap between, on the one hand, buffered terminals with terminal cluster controllers and line control in the network models, and, on the other hand, CPU models, file accessing, and front-end processor analysis in the host models. The models presented here can be used to determine bottlenecks in the entire system and to facilitate the initial phase of system design and configuration.

## Polling cycle analysis

*Polling and operations in a polling cycle*

Under normal operating conditions several terminals as well as several TCUs may be prepared to transmit transactions at the same time from remote locations to the host site. Only one can do so, and the others must wait their turns. To organize this, the line will normally be polled. For cases where terminals are controlled by the TCUs, as assumed in our model, transactions are sent to the controller at will and accumulated there so that only the TCUs need to be polled. In other cases, terminals are polled individually. Normally the local FEP (if any) or the host processor organizes the polling. In the main memory there is a polling list telling the programs the sequence in which to poll the TCUs or terminals. The polling list and its use therefore determine the priorities with which the remote devices are scanned.

There are several major time elements that constitute a polling cycle, and these include transaction transmission time, the time for either an unsuccessful (negative) or a successful (positive) poll and the associated acknowledgment. Except for the first item, all other elements depend to a great extent on the line control procedures employed by the system. Two such procedures are considered in our model: the Binary Synchronous Communications (BSC) and the Synchronous Data Link Controls (SDLC). Without going into details (see, for example, [2] and [3]), some of the differences between these are that BSC can be used only for half-duplex transmission while SDLC can provide both the half-duplex and the full-duplex modes, together with some built-in functions to reduce communications overhead as indicated in the following diagram.

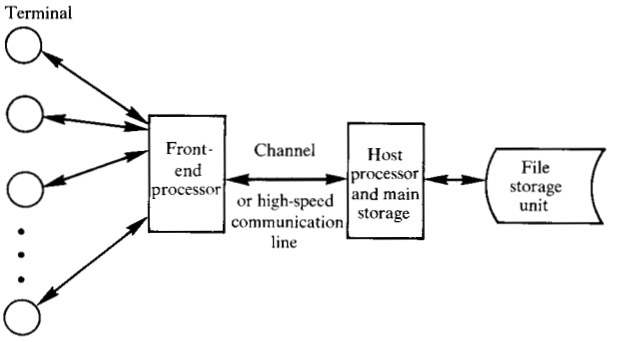
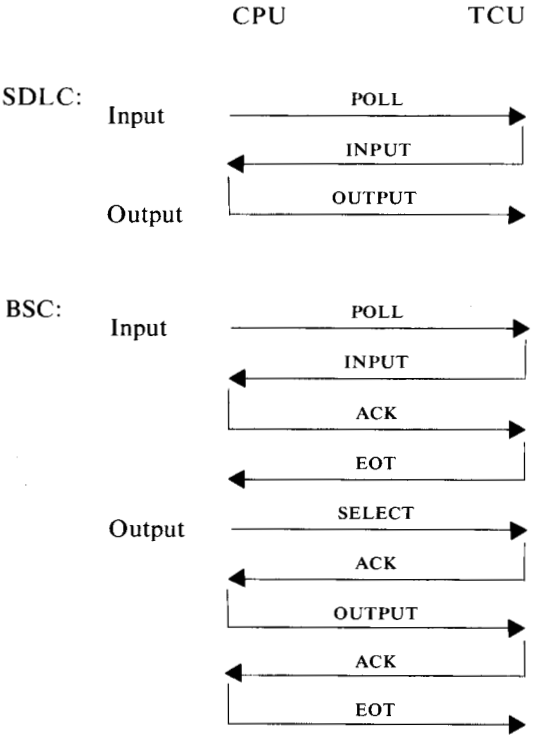



**Figure 1** A typical data-communications system.




Here POLL is the polling message generated by the host site, INPUT is the transaction transmitted from the remote terminal to the host site, OUTPUT is the transaction transmitted from the host site to the remote terminal, ACK is the acknowledgment of the receipt of a transaction, EOT is the end of transmission, and SELECT is the selection of the proper remote device for receiving a transaction from the host site.

*Transaction transmission time*

Let $K$ be a discrete random variable denoting the number of transactions removed from a typical TCU each

time this TCU is polled. The distribution function of $K$ will be considered in the next section. We also let $L_{in}$ be the length (in number of characters) of each input transaction and $S_L$ be the line speed (in number of characters per second). If there are $M_c$ TCUs in the system, the total time required for the transaction transmission during a polling cycle is then

$$t_p^{(1)} = M_c K L_{in}/S_L. \tag{1}$$

Although it is generally true that the sum of random variables may behave quite differently from each member of the sum, for analytic as well as practical reasons, we assume in Eq. (1) that all of the $M_c$ terminal control units are identical in structure and all terminals generate similar traffic. The extension to more rigorous cases is straightforward but would introduce many involved complications in computation.

*Time element related to communications overhead*

Let $L_p$, $L_a$, $L_e$ be the lengths of polling, acknowledge and EOT messages, respectively. The time element related to a positive polling is then

$$t_p^{(2)} = \begin{cases} M_c(L_p/S_L + C_t), \text{ for SDLC} \\ M_c[(L_p + L_a + L_e)/S_L + C_t], \text{ for BSC,} \end{cases} \tag{2}$$

where $C_t$ is a constant representing the time caused by modem establishment and other propagation delays.

In general the cycle time can be described in the form

$$t_p \triangleq g(K) = t_p^{(1)} + t_p^{(2)} = \begin{cases} aK + b \text{ if } K > 0 \\ C_{np} \quad \text{otherwise,} \end{cases} \tag{3}$$

where $a = M_c L_{in}/S_L$ and $b = t_p^{(2)}$, and $C_{np}$ is the time element associated with a negative poll. It is now necessary to obtain the distribution function of $K$.

• *Input process to the TCU*

For the terminal subsystem under consideration, we assume that $M_c$ TCUs are polled by a single FEP (there may be more than one FEP in large systems) and on the average $M_t$ terminals are connected to a nearby TCU. Consider a particular terminal, one out of a group with a total of $M_t$ terminals (the population source). In most interactive systems the terminal operator does not send any inquiry before the response to the previous one has been received. Thus this terminal, after a time $\tau_1$, starts to transmit a transaction for the first time. After receiving the response, the terminal becomes idle for a time $\tau_2$ before making the second request for data transmission. In general, it stays for a time $\tau_i$ in the source before making the $i$th demand for the use of communications facilities.

Let the distribution function of the inter-arrival time of transactions at a TCU be

$$A(t) = \text{Prob}\,(\tau_i \leq t). \tag{4}$$

Instead of specifying the input process through the inter-arrival-time distribution at the TCU from all the sources, which involves both the distribution and the size of the source, we specify the input process through the distribution of $\tau_i$ above, which is the inter-arrival time from one terminal.

Because the size of the source is $M_t$, the distribution of $K(t)$, the number of arrivals up to time $t$ is

$$\text{Prob}\{K(t) = k\} = \binom{M_t}{k}[A(t)]^k[1 - A(t)]^{M_t - k}, \tag{5}$$

which reduces to

$$\text{Prob}\{K(t) = k\} = \binom{M_t}{k}[1 - e^{-\lambda t}]^k e^{-\lambda(M_t - k)t} \tag{6}$$

if the arrival process is exponentially distributed with parameter $\lambda$.

• *Number of transaction removed per poll in each TCU*

The probability that $k$ transactions have arrived at a TCU during a polling cycle $t_p$ is

$$P_k = \int_0^\infty P(K(t_p) = k) f(t_p) dt_p, \tag{7}$$

as $t_p$ is a continuous random variable having probability density function (p.d.f.) $f(t_p)$. On the other hand, $t_p$ is also a function of the discrete random variable $k$ as seen in Eq. (3). It is thus proper to rewrite

$$P_k = \binom{M_t}{k} \sum_{l=0}^{M_t} \{[1 - e^{-\lambda g(l)}]^k [e^{-\lambda g(l)}]^{M_t - k} P_l,$$

$$k = 0, 1, 2, \cdots, M_t, \tag{8}$$

$$\text{and} \sum_{k=0}^{M_t} P_k = 1,$$

where $P_l$ has the same meaning as $P_k$ with the subscript changed. We now have a set of simultaneous equations

$$P_k = \sum_{l=0}^{M_t} p_{lk} P_l$$

$$\sum_{l=0}^{M_t} P_l = 1 \tag{9}$$

with the coefficient $p_{lk}$ given by

$$p_k = \binom{M_t}{k}[1 - e^{\lambda g(l)}]^k [e^{-\lambda g(l)}]^{M_t - k}. \tag{10}$$

It is noted that

$$\sum_{k=0}^{M_t} p_{lk} = 1 \text{ for all } l,$$

so that these coefficients are transition probabilities.

The solution to Eq. (9) is given below by elementary manipulations

$$(P_0, P_1, \cdots, P_{M_t}) = \frac{(1, P_1', P_2', \cdots, P_{M_t}')}{(1 + P_1' + P_2' + \cdots + P_{M_t}')}, \tag{11}$$

where

$$(P_1', P_2', \cdots, P_{M_t}') = (p_{01}, p_{02}, \cdots, p_{0M_t})[\mathbf{I} - (p_{lk}')]^{-1},$$

and where $\mathbf{I}$ is the identity matrix and $(p_{lk}')$ is the $M_t \times M_t$ matrix formed by deleting the first row and first column of the original matrix $(p_{lk})$.

After having obtained the $P_k$, the $n$th moment of the polling cycle time can readily be expressed as

$$\mathrm{E}(t_p^n) = \sum_{k=0}^{M_t} P_k[g(k)]^n. \tag{12}$$

One can now fit $f(t_p)$ by a gamma distribution function

$$f(t_p) = \frac{\beta(\beta t_p)^{\alpha-1} e^{-\beta t_p}}{\Gamma(\alpha)}, \tag{13}$$

with its Laplace transform

$$\Phi(s) = \int_0^\infty e^{-st} p f(t_p) dt_p = \left(\frac{\beta}{\beta + s}\right)^\alpha, \tag{14}$$

where

$$\alpha = \frac{\mathrm{E}^2(t_p)}{\mathrm{Var}(t_p)} \text{ and} \tag{15}$$

$$\beta = \frac{\mathrm{E}(t_p)}{\mathrm{Var}(t_p)}. \tag{16}$$

If $f(t_p)$ does not fit a gamma function, we can approximate $\Phi(s)$ by [4]

$$\sum_{r=0}^{\infty} (-1)^r \frac{S^r}{r!} \mathrm{E}(t_p^r).$$

### Waiting time for polling

The terminals in our model are assumed to be identical with respect to transaction generation intensity. The host processor (actually its local FEP) receives transactions and polls each TCU in a prescribed cyclic order (polling list). Transactions that have been keyed in and are waiting at a given TCU are transmitted almost simultaneously after this TCU is polled.

We fix our attention upon a given simple terminal (out of the whole subsystem of $M_t$ identical terminals) possessing a transaction generation intensity in number of transactions per unit of time and follow its history over a complete polling cycle. After the TCU is polled by the host processor, transactions waiting in the TCU are transmitted to the host site and the next TCU in sequence is served similarly. This particular TCU under consideration will be polled again after a random time $t_p$ (polling cycle) and the host processor may find it either empty (negative polling) or with transactions waiting (positive polling).

Because the polling signal comes to any particular TCU every $t_p$ seconds, where $t_p$ assumes some known distribution, the polling process is indeed a renewal process. In particular, it is an ordinary renewal process because the $t_p$s are independent identically distributed random variables. With the aid of some useful results available in the theory of the renewal process [4], we now evaluate the density function of the time that a transaction has to wait before being polled by the host site. This situation is similar to a queuing process in which service is available only at service-intervals, which form a renewal process. A customer arriving at time $t$ will have to wait a time $t_w$ for the first service-instant. The limiting distribution of $t_w$ can be expressed as

$$g(t_w) = \frac{1}{\mathrm{E}(t_p)} \int_{t_w}^\infty f(t_p) dt_p. \tag{17}$$

The moments of the limiting distribution of this waiting time are easily obtained from the Laplace transform. Because

$$\mathscr{L}\{f(t_p); s\} = \Phi(s) = \int_0^\infty e^{-st_p} f(t_p) dt_p = \sum_{j=0}^\infty (-1)^j \frac{s^j}{j!} \mathrm{E}(t_p^j),$$

and

$$\mathscr{L}\{g(t_w); s\} = \mathscr{L}\left\{\int_x^\infty f(u) du; s\right\} = \frac{1 - \Phi(s)}{s}, \tag{18}$$

we have

$$\mathscr{L}\{g(t_w); s\} = \sum_{j=0}^\infty \frac{(-s)^j}{j!} \frac{\mathrm{E}(t_p^{j+1})}{(j+1)\,\mathrm{E}(t_p)}. \tag{19}$$

The $j$th moment of $t_w$ about the origin, as it exits, is given by the coefficient of $(-s)^j/j!$ in the Taylor series expansion of its Laplace transform. Therefore

$$\mathrm{E}(t_w^j) = \frac{1}{j+1} \frac{\mathrm{E}(t_p^{j+1})}{\mathrm{E}(t_p)}. \tag{20}$$

Three examples of different polling-cycle time distributions are considered here:

1. The polling-cycle time is exponentially distributed with

$$f(t_p) = \mu e^{-\mu t_p},$$

where $\mu = 1/\mathrm{E}(t_p)$.

The waiting-time distribution is easily shown to be

$$g(t_w) = \mu \int_{t_w}^{\infty} \mu e^{-\mu t_p} dt_p = \mu e^{-\mu t_w},$$

and thus

$$\mathrm{E}\ (t_w) = \frac{1}{\mu}, \tag{21}$$

$$\mathrm{Var}\ (t_w) = \frac{1}{\mu^2}. \tag{22}$$

2. The polling-cycle time is a constant. The density function is then the $\delta$-function

$$f(t_p) = \delta(t_p - \frac{1}{\mu}).$$

Thus

$$g(t_w) = \begin{cases} \mu & (0 \leq t_w \leq \frac{1}{\mu}) \\ 0 & (\frac{1}{\mu} < t_w), \end{cases}$$

and

$$\mathrm{E}\ (t_w) = \frac{1}{2\mu} = \tfrac{1}{2}\text{cycle time;} \tag{23}$$

$$\mathrm{Var}\ (t_w) = \frac{1}{12\mu^2}. \tag{24}$$

3. The polling-cycle time possesses a gamma function density as given by Eq. (13). Thus

$$g(t_w) = \frac{1}{\alpha} \sum_{k=1}^{\alpha} \frac{\beta(\beta t_w)^{\alpha-1} e^{-\beta t_w}}{\Gamma(\alpha)}, \text{ if } \alpha = \text{integer}, \tag{25}$$

or

$$g(t_w) = \frac{1}{\alpha} \sum_{k=0}^{\infty} \frac{\beta^{\alpha-k} t_w^{\alpha-k-1}}{\Gamma(\alpha-k)} e^{-\beta t_w}, \text{ if } \alpha \neq \text{integer}, \tag{26}$$

with the mean and variance given by

$$\mathrm{E}\ (t_w) = \frac{1}{2} \frac{1}{\alpha^2 \mu} \frac{\Gamma(\alpha+2)}{\Gamma(\alpha)}, \tag{27}$$

$$\mathrm{Var}\ (t_w) = \frac{1}{\alpha^4 \mu^2 \Gamma(\alpha)} \left\{ \frac{\alpha}{3} \Gamma(\alpha+3) - \frac{1}{4} \frac{\Gamma^2(\alpha+2)}{\Gamma(\alpha)} \right\}. \tag{28}$$

## Delays in the front-end processor

A generic FEP can be regarded as a communications-oriented computer containing a main storage, a central control unit, a channel adapter for attachment to its host processor (a more powerful general-purpose computer) or a line adapter when used at remote locations to communicate with a local FEP, a communications line scanner, and the necessary hardware to connect a certain number of communications lines. Depending on the user's requirements, there may be several types of scanners and adapters available, differing in performance capability and cost.

Some simple transmission control units perform functions such as control character recognition, line-time-out control, error checking, and character assembly and disassembly. An FEP as described above can perform a variety of other functions such as polling and addressing of remote devices, control character insertion and deletion, character code translation, buffering, error recording and diagnosis, and the block processing capability that can correct text incorrectly entered from a station. For the purpose of analysis, all these functions can be consolidated to form three types of tasks that require service from the processor at different priority levels. Another model is required to study the channel or line adapter.

### • *Central control unit*

The Central Control Unit is essentially a processor handling three types of tasks at different priority levels. If we treat the processor as a server in the context of queuing theory, the first type of input with priority 1 (the highest) is the bit and character service in the scanner. The second job stream with priority 2 is the data transfer through the channel adapter, and the third input is the background processing in units of blocks. Each of these three types of jobs requires a certain amount of service from the processor.

*Inputs to the processor*

Let us consider a teleprocessing network of $L$ line groups with identical traffic statistics for each line belonging to the same group. There are $L_i$ lines in the $i$th group ($i = 1, 2, \cdots L$) with line utilization $\rho_i$, line speed $S_i$ (characters/second), and block length $B_i$ (characters/block).

The total network loading or the request rate of the scanner character service is

$$\lambda_1 = \sum_{i=1}^{L} \rho_i S_i L_i \text{ (ch/s)}. \tag{29}$$

Note that the time to serve a character depends on the type of scanners. Under normal conditions the output rate of the FEP should be nearly equal to the input rate, and the character service request rate at the channel adapter is

$$\lambda_2 = \lambda_1 + \varepsilon, \tag{30}$$

where $\varepsilon$ denotes the transfer of command and control information. The request rate of the background processing expressed in blocks per second is

$$\lambda_3 = \sum_{i=1}^{L} \frac{\rho_i S_i L_i}{B_i}, \tag{31}$$

and the average time to process a block $b_{13}$, is assumed to be independent of the block size. We shall use $b_{ij}$ to denote the $i$th moment of service time for the $j$th class arrivals.

*Assumptions*

In view of the fact that only average values are generally available for the arrival and service rates, it appears necessary and proper to make the following assumptions:

a. All of the three types of arrival patterns have Poisson distributions with parameters $\lambda_1$, $\lambda_2$, and $\lambda_3$, respectively.
b. The service mechanism is of mixed priority schemes in the sense that classes 1 and 2 can interrupt class 3 on a preemptive basis. The job unit for both the scanner and the adapter is in characters which are not supposed to be broken into bits before being interrupted. The job unit for background processing is in blocks consisting of many characters.

*Message delay and background queue length*

The message delay caused by the presence of the FEP and the storage required to handle the background processing are, respectively, the queuing time and queue length of the lowest priority job stream. As shown in [5], we have the mean message delay

$$T_{q3} = \frac{1}{1-u_2}\left[b_{13} + \frac{\sum_{i=1}^{3} \lambda_i b_{2i}}{2(1-u_3)}\right], \tag{32}$$

and the mean background queue length

$$Q_3 = \lambda_3 T_{q3}, \tag{33}$$

where

$$u_j = \sum_{i=1}^{j} \lambda_i b_{1i} \text{ for } j = 1, 2, 3 \text{ and } u_0 = 0.$$

• *Channel adapter*

The service provided by the FEP control program on data transfer across the channel adapter has been included in the throughput analysis. Here we shall treat the adapter itself as a server and determine the queue sizes and waiting time both at the FEP and at the host processor. The channel operation suggests that we can formulate the present system as a queuing model with two queues attended by a single server and with alternating priorities.

Let $R$ be the ratio of output rate to input rate. $R$ depends on the applications under study and is generally equal to 1 for message switching, greater than 1 for broadcasting, less than 1 for data collection. The character arrival rate at the host processor queue is then

$$\lambda_H = \frac{R}{1+R}\lambda_2 \tag{34}$$

and the character arrival rate at the FEP queue is

$$\lambda_F = \frac{1}{1+R}\lambda_2, \tag{35}$$

where $\lambda_2$ is the total network loading or throughput to the system given by Eq. (30) and subscripts H and F denote host and front-end processors, respectively.

Let the first and second moments of the service time (provided by the channel) be denoted by $b_1$ and $b_2$, respectively. As shown in [6] the average channel queuing time in the FEP is

$$t_{qF} = c_1 + a_2/2a_1, \tag{36}$$

where

$$c_1 = \frac{\lambda_F b_2}{2(1-\lambda_F b_1)},$$

$$a_1 = \frac{1-\lambda_F b_1}{1-(\lambda_F+\lambda_H)b_1}, \text{ and}$$

$$a_2 = \frac{\lambda_H b_2 (1-\lambda_F b_1)^2 + \lambda_F c_2 (\lambda_H b_1)}{[1-(\lambda_F+\lambda_H)b_1][(1-\lambda_F b_1)^2(1-\lambda_H b_1)^2 - (\lambda_F \lambda_H c_1^{\,2})^2]}.$$

Although higher moments can be determined in principle, the increment to the results is inconsequential, especially in terms of the tedious computations involved and additional information required.

## Analysis of the host processor

The major software structure of the host processor consists of many blocks of user written logic, called programs. The most important series of programs to assist the user in overall computer operation is called an operating system. The received data transmitted from the terminal is first handled by portions of the operating system and then passed to the user's application programs. Application programs are designed and implemented by the user to perform the necessary processing required in handling every business function. Access methods for teleprocessing provide an interface between the FEP and the user's application programs. Thus, the operating system, application programs, and the access methods are the three essential software parts of a host processor.

As far as the hardware structure is concerned, the use of auxiliary storage devices like disk files is essential because the main memory is limited in space. User's data, application programs, and even a major portion of the operating system are stored in these devices and are retrieved as the need arises. It is now clear that a de-

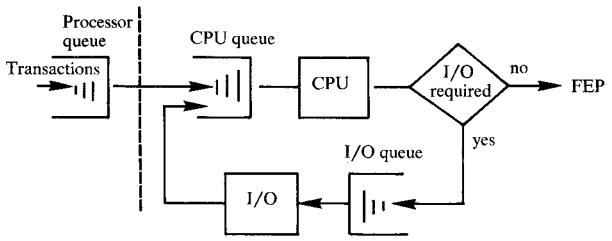



**Figure 2** A model for the host processor.

tailed analysis of the host processor is extremely complicated and requires complete information on the system (both hardware and software) and the user's applications. Because the primary goal of this paper is to study the general situation rather than to serve as a reference for any particular system or application, we analyze a model common to most systems and one that can be easily modified to meet other specific applications.

The host-processor model considered here consists of a single central processing unit (CPU), a main memory, several channels, and disk files. The disk subsystem has the feature of rotational position sensing (RPS), by which the channel and storage control are allowed to be released during most of the record search time, thus increasing channel and control unit availability for other operations. The processor itself is operated in a multiprogramming environment and assumed to have more than one program waiting for processing. The CPU makes an input/output (I/O) request whenever the program being processed issues an I/O command (such as READ, WRITE GET, PUT) either for data or for any information not available in the main memory. Having initiated this request, the CPU starts to process the next program waiting at the CPU queue. At the same time, the requested I/O operations are performed independent of the CPU. When the desired information has been transferred into the main memory, the program that made such a request will now join the CPU queue waiting for processing. This process is repeated until all the required processing is completed for a particular program and the next program waiting at the processor queue is admitted into such a partition to start these repeated operations. A schematic diagram is given in Fig. 2 to show the relationships between various queues and system components.

The basic model just described belongs to the general class of queuing networks (see [7, 8]). A special case of the queuing network is a cyclic queue that was studied by Chen and Shedler [9] for the case of exponential CPU processing time and constant I/O service. Shelder [10] subsequently allowed general distributions for the I/O service time. Expressions related to system throughput were obtained by Shedler [10] and those related to disk files were derived by Chang and Gorenstein [11] in a more general environment. Justification for and effects of various assumptions have been given in the cited references and will not be repeated here.

• *Service time of the host processor*

Consider the following assumptions:

1. There is more than one program resident in the main memory, giving rise to contention among processing resources.
2. The CPU can be operated concurrently with the information transfer unit (ITU), which consists of the channel, the control unit, and the disk devices.
3. Both the queue in front of the CPU and the queue in front of the ITU are served under a FIFO (first-in, first-out) queuing discipline.
4. System overhead is negligible.
5. $N_{pr}$, the number of programs being processed in the main memory, is a constant so that the system is in a saturated mode.
6. The successive ITU service times are independently and identically distributed as a random variable $W$ with arbitrary distribution $F_W(t)$, i.e.

$$F_W(t) = \text{Prob}\{W \leq t\} \tag{37}$$

7. The successive CPU service times are independently and identically distributed as a random variable $U$ with exponential distribution having rate parameter $u$, i.e.,

$$F_U(t) = \text{Prob}\ \{U \leq t\} = 1 - e^{-ut} \text{ for } t \geq 0. \tag{38}$$

8. A program requires a random number $M_p$ of CPU services for completion and $M_p$ has a geometric distribution with parameter $q$. The probability of termination after the $j$th CPU service is

$$\text{Prob}\{M_p = j\} = (1-q)^{j-1}q,\ j \geq 1.$$

If the above assumptions hold, then, as shown in [10], the rate of departure from the host processor has the long-term expectation

$$\text{E}(D) = \frac{q(1+\pi_0)u}{\pi_0 + u(1-\pi_0)\text{E}(w)}. \tag{39}$$

The effective service time of the processor is then

$$\text{E}(S_H) = \frac{1}{\text{E}(D)} = \frac{\pi_0 + u(1-\pi_0)\text{E}(w)}{q(1-\pi_0)u}. \tag{40}$$

Statistics of the ITU service time have been derived in [10] and $\pi_0$ is calculated by

$$\pi_0 = \left[\sum_{i=1}^{N_{pr}-1} \gamma_i\right]^{-1}, \tag{41}$$

where

$\gamma_0 = 1,$

$\gamma_1 = \gamma_0 / G_0,$

$$\gamma_i = \frac{1}{G_0}\left[(1 - G_1)\gamma_{i-1} - G_2\gamma_{i-2} - \cdots - G_{i-1}\gamma_1\right],$$

and

$$G_k = \int_0^\infty [F_k(t) - F_{k+1}(t)]\, dF_W(t); \tag{42}$$

$k = 0, 1, 2, \cdots, N_{pr} - 2$, and

$$F_k(t) = 1 - \sum_{l=0}^{k-1} \frac{e^{-ut}(ut)^l}{l!}. \tag{43}$$

Suppose now that the ITU service time has the Erlangian distribution, or

$$f_W(t) = \frac{\beta_1}{(\alpha_1 - 1)!}(\beta_1 t)^{\alpha_1 - 1} e^{-\beta_1 t}, \tag{44}$$

with $\alpha_1 = \dfrac{\mathrm{E}^2(W)}{\mathrm{Var}(W)}$ (integer part);

$$\beta_1 = \frac{\mathrm{E}(W)}{\mathrm{Var}(W)}$$

since

$$F_k(t) - F_{k+1}(t) = \frac{e^{-ut}(ut)^k}{k!},$$

and

$$\int_0^\infty t^b e^{-at}\, dt = \frac{b!}{a^{b+1}}.$$

We can show that after some algebraic manipulations

$$G_k = \frac{\beta_1^{\alpha_1} u^k}{(u + \beta_1)^{k+\alpha_1}} \binom{k + \alpha_1 - 1}{k}. \tag{45}$$

• *Response time at the host processor*

We can now calculate the response time at the host processor by treating the CPU and ITU together as a service facility. Because many transactions enter and leave the host processor, dependence between various transactions seems small and the processor service time can be assumed to have exponential distribution with parameter $\mathrm{E}(D)$, given by Eq. (39). Similar assumptions have been used by authors in studying the ARPA networks [12] and verified satisfactorily by measurements and simulations. By applying an M/M/1 (Poisson arrival/Exponential service time/Simple serve) queuing model, we can get our estimate for the processor response time given by

$$\mathrm{E}(T_H) = \frac{\mathrm{E}(S_H)}{1 - \rho_H}, \text{ and} \tag{46}$$

$$\mathrm{Var}(T_H) = \frac{\mathrm{E}^2(S_H)}{1 - \rho_H}, \tag{47}$$

where $\rho_H = \Lambda_H \mathrm{E}(S_H)$ and $\Lambda_H$ is the total transaction arrival rate at the host processor.

**Other time elements and the total response time**

So far we have calculated three major elements of a terminal response time; namely, waiting time for polling, delays in the FEP, and the turn-around time at the host processor. Other elements are the input and output transaction transmission times and the addressing time, which is just the time an output transaction has to wait for the availability of output lines.

Let $L_{\text{in}}$ be the length of input (from terminal to CPU) transaction, and $L_{\text{out}}$ that of the output transaction. If the line speed is $S_L$, then the transaction transmission times are $T_{\text{in}} = L_{\text{in}}/S_L$ and $T_{\text{out}} = L_{\text{out}}/S_L$.

Since $S_L$ is a constant, we have

$$\mathrm{E}(T_{\text{in}}) = \mathrm{E}(L_{\text{in}})/S_L,$$

$$\mathrm{E}(T_{\text{out}}) = \mathrm{E}(L_{\text{out}})/S_L,$$

$$\mathrm{Var}(T_{\text{in}}) = \mathrm{Var}(L_{\text{in}})/S_L^2, \text{ and}$$

$$\mathrm{Var}(T_{\text{out}}) = \mathrm{Var}(L_{\text{out}})/S_L^2.$$

in terms of the mean and variance of transaction lengths.

When all the required processing is finished at the host processor, the response or the output transaction is then forwarded to the FEP for transmission. Sometimes output transactions are stored in disk files before being retrieved for transaction. We now assume that these transactions are waiting in the main storage of the FEP and calculate the waiting time. Note that in the half-duplex mode the output transactions will have priority over the input transactions on a non-preemptive basis, while in the full-duplex mode the usual FIFO queuing discipline is employed. The line connecting the FEP and TCU plays the role of server in the context of queuing theory and its speed becomes the service rate. By applying an M/G/1 queuing model we have:

1. in half-duplex mode,

$$\mathrm{E}(T_a) = \frac{(\Lambda_1 + \Lambda_2)\mathrm{E}^2(S_L)}{1 - (\Lambda_1 + \Lambda_2)\mathrm{E}(S_L)}, \tag{49}$$

$$\mathrm{E}(T_a^2) = \frac{2(\Lambda_1 + \Lambda_2)\mathrm{E}^3(S_L)}{3[1 - (\Lambda_1 + \Lambda_2)\mathrm{E}(S_L)]} + \frac{(\Lambda_1 + \Lambda_2)\Lambda_2 \mathrm{E}^2(S_L)}{[1 - (\Lambda_1 + \Lambda_2)\mathrm{E}(S_L)]^2}, \tag{50}$$

**Table 1** Description of a host processor.

| | |
|---|---|
| MIPS number | 0.150 |
| Operating System | OS |
| No. of Byte Multiplexor Channels | 1 |
| No. of Block Multiplexor Channels | 1 |
| No. of Selector Channels | 1 |
| No. of FEPs | 2 |
| Degree of multiprogramming | 2 |
| Use of DASD with RPS | 50% |
| Use of DASD without RPS | 50% |

**Table 2** Description of disk files.

| | |
|---|---|
| DASD with RPS | |
| No. of modules | 4 |
| Data transfer rate | 806 kB/s |
| Rotational delay | 16.7 ms |
| No. of page-size sections | 4 |
| Constant 1 | 25 ms |
| Constant 2 | 1.57 ms |
| Total no. of track/cylinders | 100 |
| DASD without RPS | |
| No. of modules | 6 |
| Data transfer rate | 312 kB/s |
| Average seek time | 165 ms |
| Variance of seek time | 1044 |
| Rotational delay | 33.3 ms |
| Data transfer time | 10 ms |
| Overhead per record | 1 ms |

**Table 3** Communication controls.

| | |
|---|---|
| Communications access method | BTAM |
| End-of-poll list pause | 200 ms |
| Line control procedure | BSC |
| Polling message length | 7 ch |
| Acknowledgment | 2 ch |
| EOT | 2 ch |
| Time for negative polling | 700 ms |
| Time for positive polling | 1.1 s |
| Time for addressing | 1.2 s |

**Table 4** Description of line groups.

| | |
|---|---|
| Line type | HDPX |
| No. of lines/group | 3 |
| No. of terminals/line | 5 |
| Line speed in ch/s | 15 |
| Average block size | 50 ch |
| Modem turn-around time | 100 ms |
| Message ratio of output/input | 1 |
| Input message rate per terminal | 0.001 |
| Average input message length | 100 ch |
| Variance of input message length | 200 |
| Average output message length | 100 ch |
| Variance of output message length | 200 |

2. in full-duplex mode,

$$E(T_a) = \frac{\Lambda_2 E^2(S_L)}{2[1 - \Lambda_2 E(S_L)]}, \quad (51)$$

$$E(T_a^2) = \frac{\Lambda_2 E^3(S_L)}{3[1 - \Lambda_2 E(S_L)]} + \frac{\Lambda_2 E^2(S_L)}{2[1 - \Lambda_2 E(S_L)]^2}, \quad (52)$$

and

$$\text{Var}\,(T_a) = E(T_a^2) - E^2(T_a),$$

with $\Lambda_1$ and $\Lambda_2$ the input and output transaction rates along that particular line.

Now, we add a word about the total terminal response time. It is stated in previous sections that the total response time is the sum of various components and essentially no dependence exists among them because of the high degree of mixing among various transactions. We can simply write the mean (or the variance) of the sum as the sum of individual means (or the variances).

## Example

In this section we give a simple example that illustrates some of the results of the paper. The system under consideration consists of a single host processor, two front-end processors, four line groups, and several disk modules.

The host processor is a general-purpose computer and the two front-end processors represent, respectively, a simple transmission control unit and a more sophisticated communications controller. The host processor is described by its MIPS number (million instructions executed per second), operating system (OS, DOS, etc.), number of various types of channel, degree of multiprogramming, and the percentage of using disk files with as well as without the RPS feature. Table 1 shows the characteristics of the host processor analyzed in this paper. A front-end processor is specified by indicating its type (either a simple control unit or a more complicated controller), and the number of line groups with and without terminal control units, together with some time constants required to perform certain communications functions. The characteristics of disk files are outlined in Table 2, and information related to communication controls is shown in Table 3. Table 4 gives the traffic statistics for one of the eight line groups, e.g., transaction arrival rate, line speed, message length.

The nature of input transactions or the environment in which the system operates is determined by a combination of several terminal modes such as simple inquiry, inquiry and update, data entry, and complete inquiry. Transactions operating in each one of these modes would invoke specific application programs, from which the number of CPU services and disk access rates can be predetermined. As an example, the following steps may

**Table 5** Average response time for each transaction in one line group.

| *Elements* | *Load ratio** | | | | |
|---|---|---|---|---|---|
| | 0.75 | 1.00 | 1.25 | 1.50 | 1.687 |
| Waiting for polling | 1.144 | 1.176 | 1.208 | 1.240 | 1.264 |
| Input transmission | 7.767 | 7.767 | 7.767 | 7.767 | 7.767 |
| Delay in FEP | 0.001 | 0.001 | 0.015 | 0.021 | 0.315 |
| Processing in HP | 0.484 | 0.683 | 1.093 | 2.457 | 20.829 |
| Waiting for addressing | 0.104 | 0.139 | 0.174 | 0.210 | 0.237 |
| Output transmission | 7.867 | 7.867 | 7.867 | 7.867 | 7.867 |
| Total response time | 17.365 | 17.632 | 18.123 | 19.563 | 38.314 |

*Load ratio $= \lambda/\lambda_0$, $\lambda_0$ = normal load.

be required for transactions operating in a complex inquiry mode:

1. Read message.
2. Log message.
3. Process 150 instructions.
4. Read a quick look-up type of file of perhaps 20 cylinders.
5. Process 100 to 150 instructions.
6. Read a large file covering 2 to 4 packs.
7. Read an intermediate work-type file for tracking certain changes to the systems.
8. Process 200 to 300 instructions.
9. Write message to terminal.

In our example we assume equal distribution for all three of the most popular modes; namely, simple inquiry, inquiry and update, and data entry.

With the supply of the above input parameters, we can calculate the terminal response times for each line group. Table 5 summarizes the results for one particular line group and shows how each element of the response time changes as the transaction arrival rates vary. Here $\lambda_0$ represents the normal traffic rates given by Table 4. Although the results are self-explanatory, a few words seem worth mentioning. Among all of the elements that constitute the terminal response time, some of them depend solely on the physical structures of the communication path and transactions and show no effect due to traffic variation. The input and output transaction transmission times belong to this category. Other elements, on the other hand, are direct derivatives of the queuing and congestion situations. If the traffic is light, very little delay would occur. As the amount of traffic starts to build up, transactions experience longer delays and sometimes those queuing delays may become the dominating parts of the terminal response time. For systems with low-speed lines (15 characters/s or less), the input and output transmission times constitute a major portion of the total response time for light traffic situations. The use of a small or large host processor does not affect the final result much. If the input rate is high, then each transaction spends more time on the host side and this delay can be reduced significantly by employing more disk files and/or a more powerful host processor. As part of a system design tool, techniques presented in this paper can be used iteratively to check the locations of various bottlenecks and suggest ways to improve system efficiency. By taking the cost of the system into consideration, a price/performance computation can easily be made to indicate where money should be invested for the best return.

## Conclusions

As can be seen from the techniques and their potential applications, they are not all-inclusive; that is, there are parameters of interest for which analytic techniques are not provided. By going through the actual development of most of the techniques, we can easily see that their development is very complex and time-consuming and hence it would take a lot of time and effort to develop all the necessary techniques. We believe, however, that techniques provided here furnish a framework for more specific studies. It is the purpose of this paper to give an *estimate* of the total terminal response time, but not the detailed distributions, without which the actual percentile of response time cannot be obtained. A reasonable approximation of the percentile can be obtained by assuming a normal or a gamma distribution for the total response time. We suggest that further effort be devoted to the calibration and validation process. All system designers should keep in mind the problem of how to collect the correct input data and how to validate mathematical models against actual measurements.

## Acknowledgment

The author thanks Y. C. Chen, P. E. Green, A. J. Karchere, and D. T. Tang for their encouragement and support. He also thanks H. Eisenpress and L. S. Loh for their programming assistance.

## References


1. P. E. Green and D. T. Tang, "Some Recent Developments in Teleprocessing System Optimization," *IEEE Inter-Communication Conference Record*, New York, March 1973.
2. J. L. Eisenbies, "Conventions for Digital Communication Link Design," *IBM Syst. J.* **6,** 267 (1967).
3. J. P. Gray, "Line Control Procedures," *Proc. IEEE* **60,** 1301 (1972).
4. D. R. Cox, *Renewal Theory*, Methuen Monograph, John Wiley & Sons, London, 1962.
5. B. V. Gnedenko and I. M. Kovalenko, *Introduction to Queuing Theory*, Israel Program for Scientific Translations, Jerusalem, 1968.
6. L. Takacs "Two Queues Attained by a Single Server," *Oper. Res.* **16,** 639 (1968).
7. W. J. Gordon and G. F. Newell, "Closed Queueing Systems with Exponential Servers," *Oper. Res.* **15,** 254 (1967).
8. J. R. Jackson, "Jobshop-like Queueing Systems," *Management Science* **10,** 131 (1963).
9. Y. C. Chen and G. S. Shedler, "A Cyclic Queue Network Model for Demand Paging Computer Systems," *Research Report RC-2398*, IBM Thomas J. Watson Research Center, Yorktown Heights, New York 10598, March 1969.
10. G. S. Shedler, "A Cyclic-Queue Model of a Paging Machine," *Research Report RC-2814*, IBM Thomas J. Watson Research Center, Yorktown Heights, NY 10598, March 1970.
11. J. H. Chang and S. Gorenstein, "A Disk File System Shared by Several Computers in a Teleprocessing Environment," *Proceedings of PIB International Symposium on Computer—Communications and Teletraffic*, Brooklyn, NY, April 1972, pp. 177.
12. L. Kleinrock, "Analytic and Simulation Methods in Computer Network Design," *Proc. Spring Joint Computer Conference* (AFIPS) **36,** Spartan Books, New York, 1970, p. 569.


*Received May 23, 1974*

*The author is located at IBM Corporate Headquarters, Armonk, New York 10504.*