PRIORITY STATISTICAL MULTIPLEXER DESIGN FOR SNA*/SDLC ACCESS TO A VIRTUAL-CIRCUIT PACKET NETWORK

Jin-Der Wang and Ender Ayanoglu

This paper addresses the design of a statistical multiplexer as customer-premises equipment to concentrate data from multiple input ports for access to a virtual-circuit packet network based on the LAPD protocol. It is assumed that the multiplexer will be adjacent to the front-end processor of hosts employing synchronous (and, possibly, asynchronous) data link layer protocols, with emphasis on SDLC. With an overall objective of reducing end-to-end delay, we propose and analyze a priority queue solution with three queues. In this scheme, polls from the synchronous links go to the highest-priority queue (unless messages to the same cluster controller are in the queue); priority between the other two queues is set by the urgency of the application. The priority of asynchronous traffic is based on message size and favors short messages, such as echoplex. Numerical results show that giving polls priority reduces end-to-end delay significantly—even for applications with low urgency. This is because of the reduction in polling overhead, which is valid for all messages.

Jin-Der Wang *is a member of technical staff in the Data Communications Research Department of AT&T Bell Laboratories in Middletown, New Jersey. Mr. Wang joined the company in 1985 and his areas of interest are in signal processing, coding, and data network queueing analysis. He has a B.S. and an M.S. in electrical engineering from Chiao-Tung University, Taiwan, and a Ph.D. in electrical engineering from North Carolina State University, Raleigh, North Carolina.* ***Ender Ayanoglu*** *is a member of technical staff in the Network Systems Research Department of Bell Laboratories in Holmdel, New Jersey. Mr. Ayanoglu joined AT&T in 1986 and is currently working on the analysis and design of communication networks and* (continued on page 86)

* Trademark of IBM Corporation.

Introduction

For many applications, a multiplexer provides a means to concentrate mixed traffic of multiple synchronous and asynchronous data links coming from different front-end processor (FEP) ports onto a "high-speed" access line. (See Panel 1 for a list of terms and acronyms in this paper.) In this paper, we look at multiplexers that can concentrate mixed data in order to share access to a virtual private line in a packet-switched data network. We are interested in statistical multiplexers to exploit the bursty nature of data traffic. To allow for different response-time requirements for different types of traffic, as well as to

Panel 1. Terms and Acronyms in This Paper

BSC	binary synchronous communications
CC	cluster controller
CPE	customer-premises equipment
echoplex	method of checking the accuracy of transmitted data in which received data are returned to the sending end for comparison with the original
EIA	Electronic Industries Association
FCFS	first-come, first-served
FEP	front-end processor
FIFO	first-in, first-out
ISN	Information Systems Network
LAPD	link access procedure for the D channel
POS	point of sale
PR + RR	priority plus round robin (the Datakit® VCS priority scheme)
PR + SF	priority plus segmented FIFO (also known as the ISN priority scheme)
SDLC	synchronous data link control
SNA	systems network architecture
STAT MUX	statistical multiplexer
TA	terminal adapter
TDM	time-division multiplexer
VCS	virtual circuit switch
walk time	time the server spends going between two queues

make polling more efficient in a synchronous data link control (SDLC) environment, we want to give certain messages priority over others.

We quantify the effect of giving priority to messages based on application type or size, giving priority to the polls input to the multiplexers. With these results, we can recommend permanent and user-settable priority schemes, and an SDLC-frame-based service quantum for the statistical multiplexers to be developed for virtual-circuit data-network access.

In the next section, we briefly review the environment for which statistical multiplexers are intended, namely, the virtual-circuit data network. We then describe the priority schemes we propose and present our assumptions, analysis, and numerical results. The appendixes contain a detailed derivation of the analytical results outlined for the proposed priority scheme.

The Virtual-Circuit Data Network

The network we are about to describe uses the LAPD data link layer protocol. (LAPD stands for link access procedure on the D channel.) Native protocols, such as SDLC, BSC (binary synchronous communications), or an asynchronous communications protocol, are *wrapped* inside LAPD frames for transportation over the network. In this paper, we focus on SDLC as the native protocol and concentrate on remote polling,[1] in which polls and acknowledgments are also wrapped and transported over the network. (For a general description of data link layer protocols such as BSC, SDLC, and LAPD, see Reference 2.)

An SDLC or LAPD frame consists of six fields. The two fields of one byte each at the beginning and trailing ends are called *flags*; they indentify the beginning and the end of the frame, respectively. The protocol ensures that the bit sequences for flags do not occur within the rest of the frame. The *address field* and the *control field* follow the beginning flag, and are normally one byte each. The *information field* normally occupies up to 256 bytes. Finally, the *cyclic-redundancy-check field* occupies two bytes. Wrapping the native SDLC frames inside LAPD frames consists of stripping leading and trailing flags from the SDLC frames, dividing the SDLC frame (i.e., the information field plus the address, control, and cyclic-redundancy-check fields) into short frames (typically, 40 bytes), and forming a LAPD frame by adding the LAPD address, control, and cyclic-redundancy-check fields (6 bytes). Thus, a long SDLC frame is converted into several, short LAPD frames before being transported over the network.

The network for our study is shown in Figure 1. The wrapping operation is carried out by devices called *terminal adapters* (TAs), which are located either on the

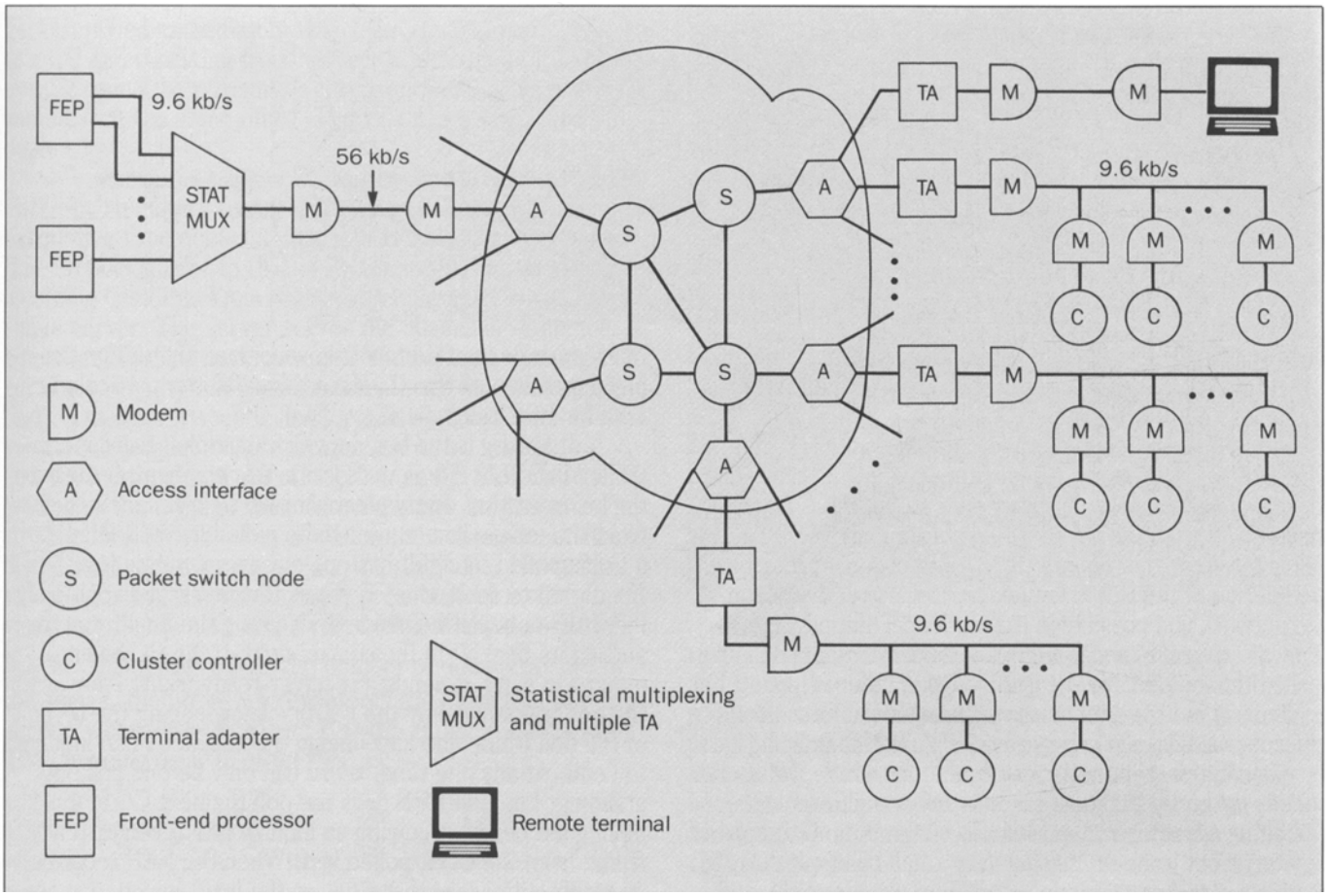


Figure 1. A typical virtual-circuit data network configuration.

customer's premises or at the network edge. When used as a statistical multiplexer, the TA is most reasonably located on the customer's premises to provide concentration over the access line. The dual operation, i.e., unwrapping, is used to form the original SDLC frames from the transported LAPD frames, and is carried out by the receive portion of the terminal adapters. All fields of the SDLC frames are treated as information within the LAPD frames.

One reason for choosing short frames for transportation over the network is to avoid the delay due to transmission of complete SDLC frames over low-speed access lines.[1] By starting to transport the leading LAPD frames before the complete SDLC frame arrives (called *pipelining into the network*), the delay due to transportation of an SDLC frame can be reduced to the transportation

Panel 2. Inquiries and Responses in Inbound and Outbound Directions

Case	Probability	Description
1	$(1 - \rho_i)(1 - \rho_r)$	No traffic in either direction (only polls and receiver-ready frames exchanged)
2	$\rho_i(1 - \rho_r)$	Inbound traffic (inquiries) only
3	$(1 - \rho_i)\rho_r$	Outbound traffic (responses) only
4	$\rho_i\rho_r$	Traffic in both directions (inquiries and responses)

time of a LAPD frame. This is about seven times shorter when the SDLC frame is of full size, assuming a LAPD frame size of 40 bytes.

At the receiving end, the SDLC frame is reconstructed and is transported over the low-speed access line. To prevent extra delay, transmission of the SDLC frame is started shortly after the first LAPD frame arrives. (This is called *pipelining out of the network.*) To prevent premature completion of the SDLC frame due to delay variations in the network and possibly in the statistical mutiplexing (known as *synchronization loss*), a deliberate short delay (called the *build-out delay*) is introduced before starting transmission. Calculations show that, for many cases, significant performance gains are possible with pipelining.[1]

In the asymmetric configuration, where the access speeds at the two sides of the network are different, the pipelining advantage is smaller. Here, the optimum strategy turns out to be (a) using a very small build-out delay in the high-speed-access to low-speed-access direction, and (b) not pipelining into the network in the low-speed-access to high-speed-access direction.

Another technique that improves performance most of the time is the use of *local polling.* In local polling, the TAs at the cluster controller (CC) side of the network are responsible for polling the CCs, and the TA at the FEP side of the network is responsible for emulating the CCs and being polled by the FEP. Local polling keeps the network free from poll traffic and increases the polling rate. One difficulty associated with local polling is that there are many variations of protocols in use; thus, the design usually depends on a case-by-case basis. This is particularly true for BSC as opposed to SDLC.

Polling is the way in which multiple access is controlled in SDLC. We now describe this mechanism for full-duplex operation. In this mechanism, CCs collect inquiries from the terminals and wait to be polled by the FEP. When a CC is polled, and if it has one or several messages (inquiries) to send, the CC sends this message in (possibly more than one) SDLC frames, denoting the final frame by setting its final bit in the control field. If the CC has no inquiry to send, it sends a receiver-ready frame (similar to the poll frame used by the FEP) acknowledging the receipt of the poll frame and announcing the absence of an inquiry to send. At any one time, there can only be one poll outstanding, i.e., the FEP does not poll the next CC in its polling list before receiving an inquiry or a receiver-ready frame from the CC it polled last. When the FEP receives an inquiry, it passes that along to the host, which may generate a *response.* The responses are queued by the FEP to be transmitted to the CC.

In the FEP, polls get nonpreemptive priority over SDLC response frames. That is, when the FEP receives either the final frame of a set of inquiries or a receiver-ready frame in response to a poll, it generates a poll for the next CC in its polling list. If there is a response being transmitted, the poll waits for completion of the current SDLC frame to be launched, but the poll gets priority over the other SDLC frames in the queue. Otherwise, the poll

gets launched immediately. The FEP can be transmitting to a CC and receiving from another at the same time; however, it cannot be transmitting to and receiving from the same CC at the same time. (The CCs are operated half-duplex.)

Multiqueue Analysis. To represent the delay experienced by an inquiry from a CC to the FEP, we use the exhaustive-type queueing model of Hashida,[3] and a "walk time" model similar to that of Reference 1. In this model, incoming customers join one of several queues served by a single server. The server serves the queues in sequence, but, after finishing service to one and before beginning service to another, does some other work (or takes a vacation). The time it spends doing other work is a random variable called the walk time, denoted by w, whose first and second moments are of interest. In the formulation leading to the average delay formula below, it is assumed that the interarrival times of customers are exponentially distributed with average λ_j^{-1} for the jth queue. Every customer demands a service time represented by a random variable τ that is not necessarily exponentially distributed. If $\lambda = \sum_{j=1}^{N} \lambda_j$, where N is the number of queues served by the server, then $\rho = \lambda \bar{\tau}$ represents the utilization of the server, where $\bar{\tau} = E[\tau]$ and E is the expectation operator. The queueing delay is given then as:[3]

$$\Delta_{mq} = \frac{(N-1)\bar{w}}{2(1-\rho)} + \frac{\overline{w^2}}{2\bar{w}} + \frac{\lambda\overline{\tau^2}}{2(1-\rho)} \qquad (1)$$

where, again, the bar symbol denotes the expectation operator.

We represent the work that is performed between serving the CCs as "walking" in the model above. We assume a response is generated for every inquiry received, and no response is generated without an inquiry so that the arrival rates of messages in the inbound direction (CC to FEP) are the same as the outbound direction (FEP to CC). However, because the average time it takes for an inquiry to be transmitted (t_e) is different than the time for a response (t_r), the utilizations of the inbound line ($\rho_i = \lambda t_e$) and the outbound line ($\rho_r = \lambda t_r$) are different. Note that $\rho_i(\rho_r)$ represents the average percentage of time (i.e., the probability) that there are messages in the inbound (outbound) line. Because of host processing, the relationship between the inbound and outbound lines is weak; we assume that message flows on these two lines are statistically independent.

We model w as a discrete random variable representing the four possible cases of the simultaneous existence or absence of inquiries and responses in the inbound and outbound directions. For example, the probability that there are no inquiries and no responses (i.e., only poll frames and receiver-ready frames are being exchanged) is given as $(1-\rho_i)(1-\rho_r)$. We use the convention shown in Panel 2 and the approximate probabilities to describe the events that this random variable represents. Panel 3 describes the symbols to be used. (Subscript f represents FEP side parameters, subscript n

Panel 3. Components of Walk Time and Delay

t_p	transmission time of a poll frame
t_{rr}	transmission time of a receiver ready frame
t_e	transmission time of an average inquiry frame
t_r	transmission time of an average response frame
t_{sdlc}	transmission time of a full SDLC frame
t_{lapd}	transmission time of a full LAPD frame
$\delta_{RS/CS}$	modem turnaround time
$\delta_{netproc}$	network processing time
δ_{prop}	propagation delay through the transmission medium
$\delta_{modem,\,f}$	modem processing delay per modem pair at the FEP side of the network
$\delta_{modem,\,c}$	modem processing delay per modem pair at the CC side of the network
ϵ	build-out delay

network side parameters, and subscript c CC side parameters.)

Consider case 1, in which there is no inquiry or response traffic in either direction. There are only poll and receiver-ready frames being exchanged, and the walk time for this event is the sum of the delays a poll and a receiver-ready frame experience in being transported over the low-speed access lines in the two ends of the network, two pairs of modems on these lines, and the network. Therefore,

$$w_1 = t_{pf} + t_{pn} + t_{pc} + t_{rrf} + t_{rrn} + t_{rrc} + \delta_{RS/CS} + 2\,\delta_{netproc} + 2\,\delta_{prop} + 2\,\delta_{modem,c} + 2\,\delta_{modem,f}$$

In case 2, the FEP is receiving inquiry frames but not transmitting responses. Because the acknowledgment in this case is indicated by setting the final bit in the control field of the final SDLC frame, there are no receiver-ready frames transmitted, and the transmission time of the receiver-ready frames should be excluded from the walk time. On the other hand, during the build-out period and the first LAPD frame transmission time, the server is idle, not doing useful work. Therefore, these times should be included in the walk time. This gives:

$$w_2 = w_1 - t_{rrf} - t_{rrn} - t_{rrc} + \epsilon_f + t_{lapd,c} + t_{lapd,n}$$

In case 3, the FEP is not receiving inquiries but is transmitting responses. In this case, the poll should wait, on the average, half of the SDLC frame transmission time for the completion of an SDLC frame. As in case 2, the time spent during build-out and while waiting for the first LAPD frame to arrive at the CC side TA is experienced by all the response frames, and is also the delay seen by a poll frame. Hence, they should be included in the walk time. Therefore,

$$w_3 = w_1 + 0.5\,t_{sdlc,f} + \epsilon_c + t_{lapd,f} + t_{lapd,n}$$

Case 4, in which there are both inquiries and responses, can be analyzed by incorporating both factors above. This gives:

$$w_4 = w_1 - t_{rrf} - t_{rrn} - t_{rrc} + \epsilon_f + t_{lapd,f} + t_{lapd,n} + 0.5\,t_{sdlc,f} + \epsilon_c + t_{lapd,c} + t_{lapd,n}$$

Finally, the first and the second moments can be found by using the probability assignments in Panel 2 and the walk time variables.

The overall delay is given as:

$$\Delta = \Delta_{mq} + t_{ec} + t_{en} + t_{ef} + \delta_{netproc} + \delta_{prop} + \delta_{modem,c} + \delta_{modem,f} + t_{lapd,f} + \epsilon_f + \Delta_r + t_{rf} + t_{rn} + t_{rc} + \delta_{netproc} + \delta_{prop} + \delta_{modem,f} + \delta_{modem,c} + t_{lapd,c} + \epsilon_c \qquad (2)$$

where Δ_r is the queueing delay at the FEP side experienced by a response frame. We assume an M/M/1 queue to model this phenomenon. This assumption gives:

$$\Delta_r = \frac{\lambda(t_{rf})^2}{(1 - \rho_r)} \qquad (3)$$

Note that in Δ_{mq} in equation (2), ρ of equation (1) should be replaced by ρ_i. If an M/M/1 queue is formed at a CC, $\overline{\tau^2}$ of the third term in equation (1) should be replaced by $2\,t_{ec}^2$. The effect of placing the TA at the FEP is reflected by multiplying t_{pf}, t_{rrf}, and t_{rf} by [1 + (6/40)] = 1.15. (Recall that we assume a 40-byte LAPD message size with 6-byte headers.) We caution the reader that, if the inquiry is less than a full LAPD frame, there is no build-out introduced at the FEP-side TA, and $\epsilon_f = 0$. In this case, the two delay values $t_{lapd,c}$ and $t_{lapd,n}$ in w_2, w_4, and equation (2) should be adjusted to the average inquiry length, rather than to the LAPD frame size. Similarly, if the response is less than a full LAPD frame, then $\epsilon_c = 0$, and $t_{lapd,f}$ and $t_{lapd,n}$ in w_3, w_4, and equation (2) should be adjusted accordingly. Fur-

ther, if the response is shorter than a full SDLC frame, $t_{sdlc,f}$ and $t_{sdlc,c}$ should be changed to the transmission time of the average response frame over the FEP-side and CC-side access line, respectively.

A Priority Scheme for Multiterminal Adapters

As mentioned before, a multiplexer provides a means to concentrate mixed traffic of multiple synchronous and asynchronous data links coming from different FEP ports onto a high-speed access line. In this paper, we consider a typical configuration connected to a virtual-circuit data network such as that in Figure 1.

Before going further, we describe this configuration. The access line speed at the FEP side is 56 kilobits per second (kb/s); at the cluster controller (CC) side, it is 9.6 kb/s. The transmission (EIA) speed at the FEP port is 9.6 kb/s. In our delay calculations to be presented, the multiplexer has either 6, 12, or 16 input ports. The trunk speed of the network is assumed to be 1.544 megabits per second (Mb/s). We are most interested in understanding the effect of various queueing delays on a message as it goes through the network. There are queueing delays in both the inbound and outbound directions. In the inbound direction, in addition to minor delays inside the network, messages could be queued at the CCs and on the FEP network edge. In the outbound direction, in addition to minor delays inside the network, the messages could be queued at the FEP and at the multiplexer output.

For our configuration, no queue exists at the TA on the CC side.

The delay bottleneck usually occurs at the multiplexer output. In particular, this is the case in the configuration we consider. Therefore, in the following sections we concentrate on analyzing the effect of multiplexer design on delay performance.

There are many possible implementations of a multiplexer. Two well-known schemes based on time domain considerations are the synchronous time-division multiplexer (TDM) and the statistical multiplexer (STAT MUX). It is well-known that statistical multiplexing is more efficient than TDM under low and medium traffic conditions (Reference 4, p. 129). Also, for a STAT MUX, the sum of the input EIA speeds can be greater than the trunk speed. Finally, it is possible for a STAT MUX to assign priority to urgent traffic, i.e., applications that are more critical in response time than others. Typical examples of "urgent traffic" include: the point of sale (POS) application that requires quick response, poll frames of synchronous data links that determine the polling overhead, and short messages of interactive applications. If local polling is used, the poll frames are not present at the STAT MUX. Our examples assume remote polling. However, the basic STAT MUX design philosophy remains the same when local polling is used, except that giving polls priority is irrelevant.

The reader should realize that analyzing such a system in full detail may be prohibitively complicated. For the delay analysis to be tractable, some simplifications need to be made. In this paper, the message delay of each data link at the FEP is calculated by assuming an M/G/1 queueing process. (An M/M/1 queue will be used in numerical examples. It is a special case of an M/G/1 queue.) The queueing (access) delay at the CC is calculated by a multiqueueing model assuming a Poisson arrival process to each CC, where each CC is assumed to have the same traffic statistics.

When appropriate, we use Kleinrock's well-known independence assumption, which is common in the queueing analysis of computer networks.[5] For example, we assume that the composite arrival process to the STAT MUX is Poisson and that the service times are independent in spite of the dependence between interarrival times and message lengths at each input of the STAT MUX (a) because of queueing at the FEP and (b) because of the fact that the message is not instantaneously available to the STAT MUX. The justification of this assumption is that the composite arrival process at the STAT MUX is actually a mixing of Poisson messages from several input lines. This, in fact, assumes that, when analyzing the STAT MUX, we ignore its dependence on the queueing delay at the FEP.

We also make this independence assumption for those queues that a message encounters inside the virtual-circuit data network. Such queues are treated as a cascade of independent M/D/1 queues.

Before describing our proposed priority scheme, we review two priority schemes used in some versions of AT&T's Datakit® virtual circuit switch (VCS). and the Information Systems Network (ISN).[6] These schemes are designed to favor short messages (i.e., interactive traffic) over long messages (file transfer). The ISN priority scheme (also known as "priority plus segmented FIFO," or PR + SF) is a two-queue implementation. Any message smaller than a service quantum goes to the high-priority queue in a first-in, first-out (FIFO) fashion. The other messages are divided into small packets, and those packets are put in a low-priority queue by consecutively taking one packet from each message, i.e., the packets are "intermingled." The low-priority queue gets served only when the high-priority queue is empty. On the other hand, each input source (whose priority scheme is known as "priority plus round robin," or PR + RR) has a dedicated data queue. Besides the data queues, there are two channel queues that store channel numbers. The FIFO channel queue lists the channel numbers of the nonempty data queues. If the message requires additional service, the channel number is listed in another channel queue. It takes successive turns in a low-priority round robin, which is served only when the FIFO channel queue is empty, in a nonpreemptive fashion. The ISN priority scheme approaches a single queue solution in which no priority is enforced when most of its messages are smaller than a service quantum.[6]

The Datakit VCS and ISN schemes are designed so that small asynchronous messages are given priority. No further a priori knowledge is assumed in their design. They do not distinguish between messages from different data links. On the other hand, we are designing a STAT MUX to be used as customer-premises equipment (CPE). Our design tries to optimize the delay performance by taking advantage of a priori knowledge of the traffic. In the environment we consider, the STAT MUX has multiple input ports, each dedicated to either a synchronous or asynchronous data link that belongs to the same customer. Because the customer has the luxury of having some a priori knowledge of the traffic, a more selective scheme can be designed to enhance the end-to-end delay performance. Our more selective priority scheme is designed in such a way that poll frames from all the synchronous data links are put in the highest-priority queue and the other messages in lower-priority queues. However, polls cannot be given priority over messages to the same CC. We will see that enforcing this priority reduces the walk time and leads to a considerable reduction in the end-to-end message transaction delay.

It is also possible to enforce priority among data links. Because messages from certain ports can be more critical in response time than others, they can be given higher priority. For example, most customers would prefer that POS messages be given higher priority than file transfers or banking applications. It is also important to have the flexibility to give priority to asynchronous messages that are smaller than a predetermined quantity. For example, we might want to give priority to echoplex. In such a scheme, the other messages are divided into small packets according to the service quantum and intermingled in a low-priority queue. We should note that the choice of a service quantum is a trade-off between the pipelining advantage and the excessive build-out delay that is needed to compensate for the time variation due to the intermingling of packets in statistical multiplexing. The pipelining at a multiplexer output is usually reduced because of the wait for a service quantum. In our case, there are two reasonable choices of a service quantum. One is a LAPD frame size, the other an SDLC frame size. Using a service quantum of a LAPD frame size would preserve the pipelining advantage to a certain extent, depending on traffic density at a statistical multiplexer; however, this would mean using a larger build-out delay. In the numerical results to be presented, we will assume that a service quantum of an SDLC frame size is used.

If the service quantum is a LAPD frame, the

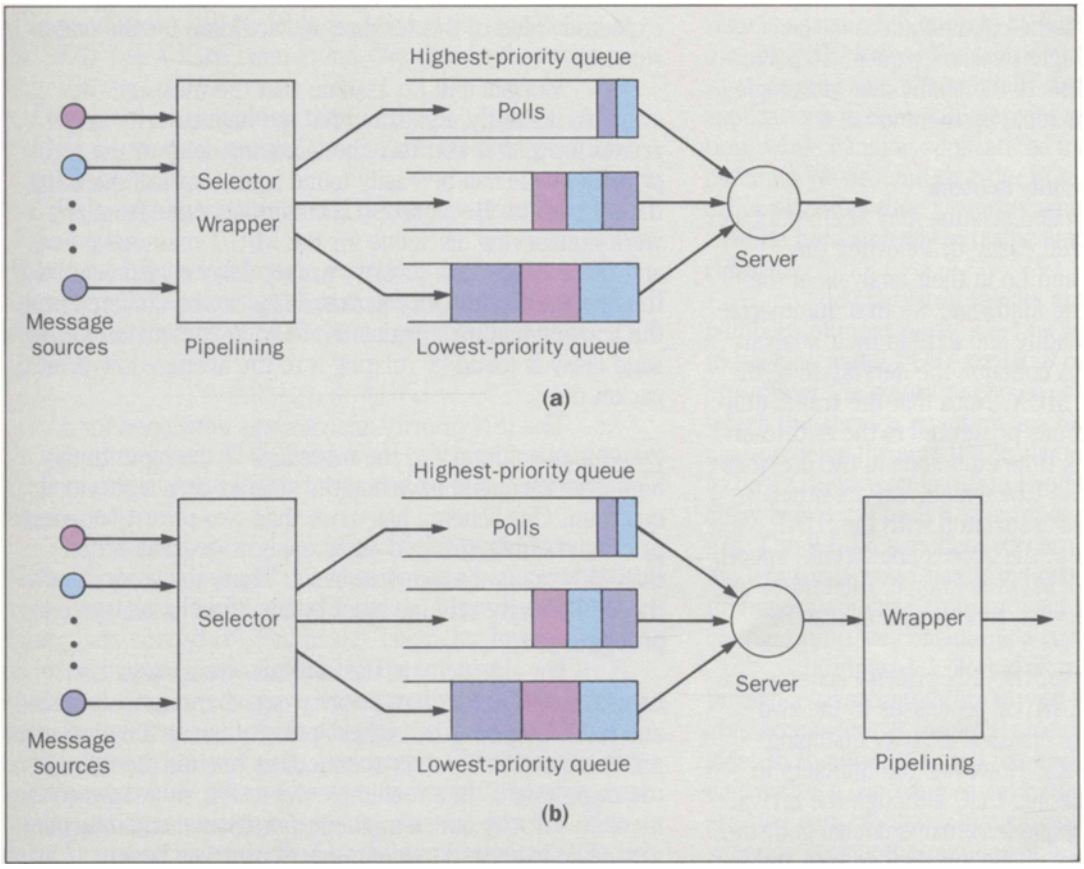


Figure 2. Proposed schemes using (a) LAPD service quantum and (b) SDLC service quantum.

wrapping function is included in the division or partitioning process as shown in Figure 2a so that the partitioning process becomes a packetization process. On the other hand, if the service quantum is an SDLC frame, the wrapping function (including pipelining into the network) follows the server as shown in Figure 2b.

It is possible to assign as many priorities as needed to classify traffic. We propose a three-queue solution that improves performance while maintaining simplicity. In this solution, the highest-priority queue is dedicated to the poll frames. The medium-priority queue is dedicated to other urgent messages such as POS. The interactive messages that are smaller than a predetermined quantity can join either the highest-priority queue or the medium-priority queue, depending on a customer's preference. The messages coming from applications such as file transfers are divided into small packets and intermingled in a low-priority queue. It is important to note that the selection criteria of our priority scheme, which is based on the urgency of the applications, is different than the scheme used in ISN. The disadvantage of the ISN priority discipline with a single-queue solution where no

priority is enforced under some circumstances is also avoided by dedicating the highest-priority queue to polls. This priority design only affects the traffic due to a single customer and can be set or reset by the end-user.

Analysis of the Proposed Priority Scheme

Although the proposed scheme may seem difficult to analyze, it can be analyzed easily by following the approach used by Morgan and Lo in their analysis of the ISN priority scheme.[6] In the following, we first summarize their approach, and then modify and extend their analysis to our problem. Appendix A contains the derivation of the average delay at the STAT MUX. Note that the traffic utilizations and transmission times presented in the end-to-end delay formulas and the walk time equations in the previous sections are evaluated at the EIA speed. On the other hand, the various quantities associated with the STAT MUX design are evaluated at the access line (trunk) speed. Once the delay at the STAT MUX is known, the end-to-end delay formulas and the walk time equations can be modified.

As mentioned above, the ISN priority scheme is a two-queue approach. If the arrival processes to the two priority queues are Poisson, classical priority queueing results can be applied directly. However, the difficulty in the analysis comes from the fact that, although the arrival process of the high-priority queue remains unchanged, the messages to the low-priority queue are divided into small packets and intermingled. Thus, the arrival process is non-Poisson.

An important contribution by Morgan and Lo in their approach is to use the well-known, but seldomly used, G/G/1 conservation law. The G/G/1 conservation law states that:

$$\sum_i \rho_i E[W_i] = E[U] - E[W_0]$$

where $E[U]$ is the expected value of the unfinished work in the system, $E[W_i]$ is the waiting time, and $E[W_0]$ is the expected value of the residual service time for the one in service.

Morgan and Lo assume that the messages are available instantly. Realizing that the high-priority queue arrival process is Poisson, the queueing delay at the high-priority queue can be easily found from classical queueing theory results. Because $E[U]$ can be obtained from any work-conserving discipline for the M/G/1 message queue, and the average high-priority packet delay is also available, the average low-priority packet delay can be obtained from the above equation. Finally, the average low-priority message delay is found by relating it to the average low-priority packet delay.

The ISN priority analysis was developed for a two-queue solution and the messages at the input to the high-priority queue have lengths smaller than a service quantum. Our scheme has more than two priority queues; priority could be granted on an application basis rather than only on a message size basis. Thus, the analysis of the ISN priority scheme must be modified to analyze our problem.

For simplicity in the analysis, we assume that messages in the medium-priority queue are not partitioned and intermingled. The highest-priority queue only has non-preemptive priority over the medium one for the full message length. In a real implementation, messages to the medium-priority queue might be partitioned and intermingled. The high- and medium-priority queues have nonpreemptive priority over the low-priority queue for a packet only (a segment of the partitioned low-priority message). Consequently, arrival processes to the medium- and high-priority queues are Poisson. This makes the analysis mathematically tractable.

Realizing that the priority scheme is a special case of the head-of-line nonpreemptive priority queue, we can readily obtain the average high-priority message delays directly from classical queueing theory results. With reasonable effort, we are able to extend the ISN priority analysis for our problem.

The mathematical details for this paper are found

in Appendixes A and B. We assume that various delays at the STAT MUX have been found. We denote these delays as:

- $E[W_h]$—average waiting time of the poll frames
- $E[W_m]$—average waiting time of the medium-priority messages
- $E[W_l]$—average waiting time of the low-priority messages.

We can modify the walk time equations and end-to-end delay formula to take these extra delays into account. The walk time equations of our scheme are modified as:

$$w_i' = w_i + E[W_h]$$

On the other hand, $E[W_m]$ and $E[W_l]$ should be added to medium-priority and low-priority end-to-end delay formulas, respectively.

We compare the calculated delay performance between our scheme and one that places all information from every data link to a single queue, served on a first-come, first-served (FCFS) basis. The delay formulas can be found in any classical queueing theory text.

As expected, our scheme improves delay performance. In the next section, to address the question of the need to give priority to polls, we compare two designs: one as proposed above, the other with two priority queues, each one accepting information from one class of data link. The delay formulas can be derived by the same approach used in Appendix A. Walk time equations and end-to-end delay formulas should be modified according to Appendix B.

Numerical Results

In the calculations to be presented, only remote polling (through-the-network polling) is considered. There are two classes of full-duplex SDLC data links. One is assumed to be a POS application. For this application, we assume (based on some field measurements) that on the average, an inbound message contains 17 bytes and an outbound message 45 bytes. For the other application, assumed to be banking, on the average, an inbound message is assumed to contain 80 bytes and an outbound message 810 bytes. We arbitrarily assume that the message lengths are exponentially distributed and that POS applications make up 25 percent of the total traffic utilization, while banking applications make up the other 75 percent. We assume that the POS data links are more critical in response time than the banking data links. We also assume homogeneity in traffic among data links of each class.

The calculated results are presented with different numbers of input ports, i.e., 6 (3 banking and 3 POS), 12 (6 banking and 6 POS), or 16 (8 banking and 8 POS), with or without the priority scheme. We consider the case in which the service quantum is a full SDLC frame. The propagation delay through the transmission medium is assumed to be 12 milliseconds (ms). The 56-kb/s modem processing delay is 5 ms. The 9.6-kb/s modem processing delay is 15 ms. The 9.6-kb/s modem turnaround time is 30 ms. The modem turnaround times and processing delays are realistic figures obtained from product specifications. The configuration we consider is a typical one.[7]

In Figures 3, 4, and 5, we show the delay performance of a configuration with 9.6-kb/s EIA speed, 56-kb/s access line at the FEP side, and 9.6 kb/s at the CC side. In Figures 3, 4, and 5a, each data link has a tail circuit with 5 CCs; whereas in Figure 5b, it is 16 CCs. There are two sets of curves in each figure that show the delay in seconds versus the number of transactions per cluster controller per hour. The two sets of curves (displayed in red and blue) show the delay comparisons with and without enforcing the priority mechanism.

As mentioned above, applications for which delay is critical are given priority. Figure 3 shows the average message delay of synchronous banking data links. One set of curves assumes an SDLC frame service quantum in which our priority scheme is enforced (curves a, c, and e, shown in red). The other set assumes a message service quantum in which no priority scheme is enforced (curves b, d, and f, shown in blue). Figure 4 shows the average message delay for synchronous POS data links. It may be

Figure 3. Average message delay of synchronous data links for a banking application with 5 CCs per data link. Red curves show banking delay performances, where polls from both banking and point-of-sale (POS) data links are given highest priority—for cases of 8 banking/8 POS (curve a), 6 banking/6 POS (curve c), and 3 banking/3 POS (curve e) at the input of the statistical multiplexer. Blue curves show performances where no priority is granted—for cases of 8 banking/8 POS (curve b), 6 banking/6 POS (curve d), and 3 banking/3 POS (curve f).

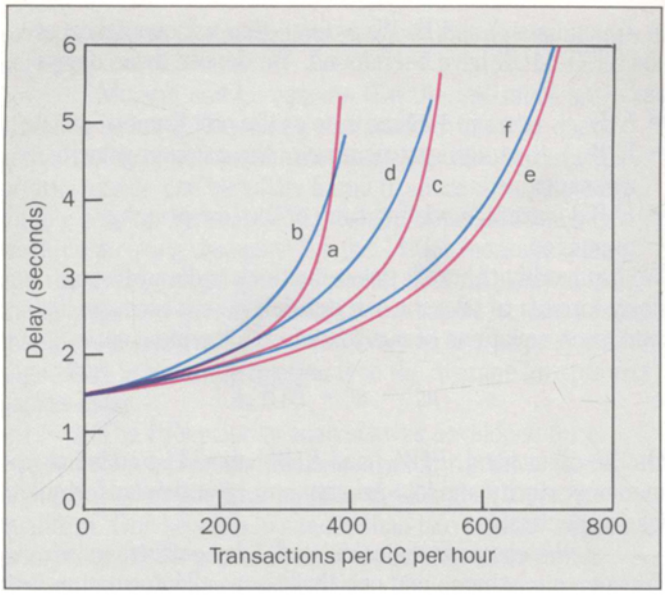
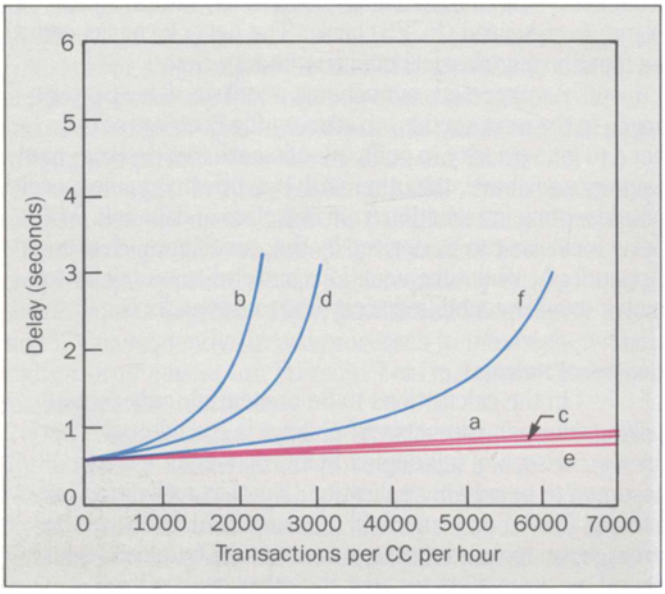


Figure 4. POS delay performance with 5 CCs per data link. Red curves show POS delay performances, where polls from both banking and POS data links are given highest priority, POS messages medium priority, and banking messages lowest priority—for cases of 8 banking/8 POS (curve a), 6 banking/6 POS (curve c), and 3 banking/3 POS (curve e) at the input of the statistical multiplexer. Blue curves represent the POS delay performance where no priority is granted—for cases of 8 banking/8 POS (curve b), 6 banking/6 POS (curve d), and 3 banking/3 POS (curve f).



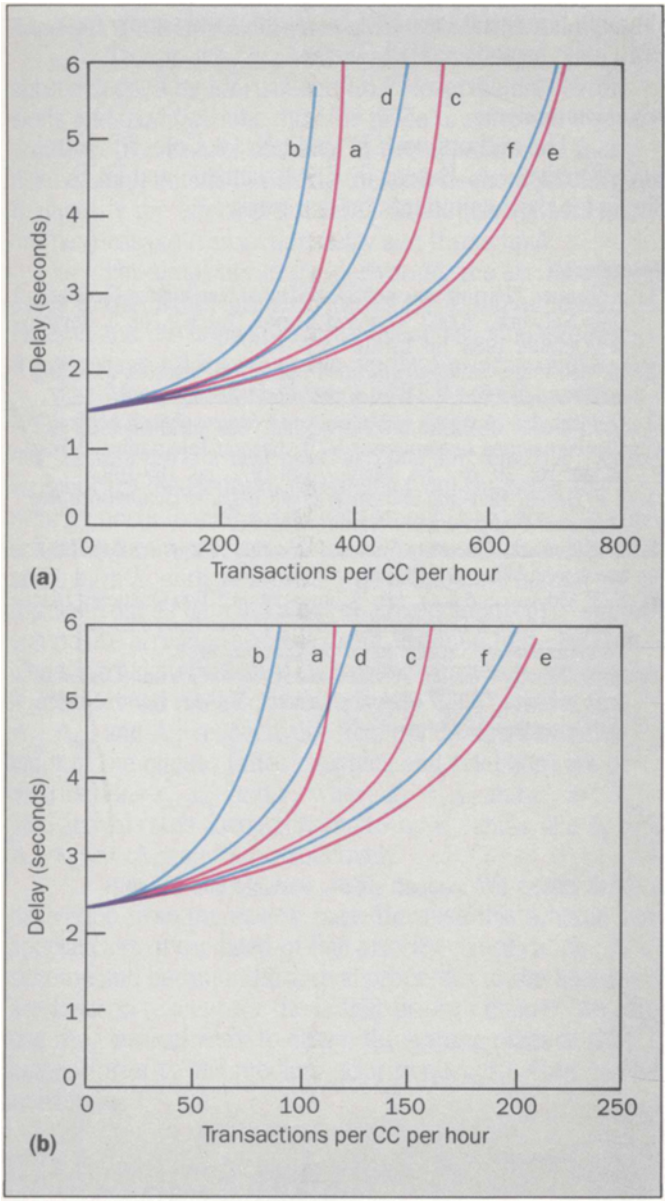


Figure 5. Banking delay performance, with and without giving polls priority, for (a) 5 CCs per data link and (b) 16 CCs per data link. Red curves represent the banking delay performances, where polls from both banking and POS data links are given highest priority, POS messages medium priority, and banking messages lowest priority—for cases of 8 banking/8 POS (curve a), 6 banking/6 POS (curve c), and 3 banking/3 POS (curve e) at the input of the statistical multiplexer. Blue curves represent performances, assuming that POS (including polls from POS data links) is given high priority and banking (including polls from banking data links) low priority—for cases of 8 banking/8 POS (curve b), 6 banking/6 POS (curve d), and 3 banking/3 POS (curve f).

expected that, by giving POS messages preference, the banking message delay will be worse than its counterpart without priority classification. Results in Figure 3 are contrary to this intuition, however. Not only does the POS delay performance improve, the banking delay performance also improves under our priority scheme. This is because of the polling overhead reduction (and thus a reduction in walk time) in giving poll frames the highest priority. Figure 4 shows that, if the POS application is not given priority, its throughput is severely limited. This result clearly shows that, if priority is not given to delay-critical applications, the throughput might not meet the customer's need. We see the smallest improvement in the case of 3 banking/3 POS because the delay at the FEP queue begins to dominate. These results resolve the question of the necessity of giving priority to delay-critical applications. Figures 5a and 3 compare the delay performances with and without giving polls priority. Figure 5a shows that there is a major improvement by giving polls priority. In fact, Figures 3 and 5a show that, if polls are not given priority, the banking message delay under priority classification is worse than the case without any priority classification. This result confirms the intuition mentioned above. We notice from calculation results that the performance of delay-critical applications in either case is

substantially improved. These results are slightly optimistic in that they assume all polls are given priority, even over messages to the same CC. Because we assume a large number of CCs, this error is small.

Figure 5b shows delay performances under the same condition as Figure 5a, except that the number of CCs is changed from 5 to 16. We see greater delay improvement in Figure 5b than in Figure 5a. This is because of the increase in the number of CCs at each tail circuit—thus, the importance of the polling overhead in the end-to-end delay calculation.

Conclusion

We have examined the design of a statistical multiplexer as customer-premises equipment that could concentrate data from multiple data input ports for access to a virtual-circuit packet network based on the LAPD protocol. It is assumed that the multiplexer would be used adjacent to the front-end processor of hosts employing synchronous (and in addition, possibly, asynchronous) data link layer protocols. With the overall objective of reducing end-to-end delay, we have proposed and analyzed a priority queue solution with three queues. We have concentrated on remote polling, but, even if local polling were used, the design philosophy remains the same, except for giving polls priority.

In this scheme, polls from the synchronous links go to the highest-priority queue (unless messages to the same CC are in queue). Priority between the other two queues is set with respect to the urgency of applications in some links (such as in point-of-sale applications over banking applications). Or, priority can be set with respect to message size among interactive traffic.

Calculation results show significant reduction in end-to-end delay by giving polls priority in both point-of-sale and banking applications even when banking applications are given the lowest priority. This is because of the reduction in polling overhead, valid for all messages. It has been observed that not giving point-of-sale applications priority severely restricts their throughput. Finally, although our results are SDLC-specific, they apply to SDLC-like protocols (HDLC, etc.).

Acknowledgments

The authors wish to thank R. G. Cole, M. Karol, T. C. Kerrigan, A. Kumar, J. B. Medamana, and D. D. Sheng for their comments on this work.

References

1. A. Kumar, "Performance of SNA/SDLC Over Virtual Circuits in a Data Network," *AT&T Technical Journal*, Vol. 67, No. 5, September/October 1988, pp. 27-40.
2. M. Schwartz, *Telecommunication Networks: Protocols, Modeling and Analysis*, Addison-Wesley, Reading, Massachusetts, 1987.
3. O. Hashida, "Analysis of Multiqueue," *Review of the Electrical Communication Laboratories, NTT*, Nippon Telegraph and Telephone, Vol. 20, No. 3-4, March-April 1972, pp. 189-199.
4. J. F. Hayes, *Modeling and Analysis of Computer Communications Networks*, Plenum Press, New York, 1986.
5. L. Kleinrock, *Queueing Systems, Volume 2: Computer Applications*, John Wiley & Sons, New York, 1976.
6. S. P. Morgan and C. Y. Lo, "Comparison of Two Queueing Disciplines for Mixed Data Traffic," *IEEE Transactions on Communications*, accepted for publication, 1989.
7. D. D. Sheng, "Virtual Private Line Performance and Customer Cost Impacts," *AT&T Technical Journal*, Vol. 67, No. 6, November/December 1988, pp. 47-68.

Appendix A. Priority Analysis of a Statistical Multiplexer

The priority analysis here follows closely the approach used by Morgan and Lo.[6] Nevertheless, extensions and modifications must be made to solve our problem. The results obtained in this appendix are then used in Appendix B to modify the end-to-end delay formula to quantify the effect of our STAT MUX priority scheme on the message transaction delay and throughput.

For simplicity in the derivation, the arrival processes to the higher-priority queues are assumed to be Poisson and the highest-priority queue has a nonpreemptive message priority over the medium-priority queue.

Messages at the output of FEP priority ports (POS data links in our case) are consolidated into a process, assumed to have Poisson arrivals and length X_m with an arbitrary distribution. Messages from the other low-priority ports (banking data links in our case) are also consolidated into a process, assumed to have Poisson arrivals and length X_l with an arbitrary distribution. The poll frames from all data links are of a fixed length, X_h, and its composite arrival process is again assumed to be Poisson. The length distribution of the overall composite process is X. The arrival rates corresponding to X_h, X_m, and X_l are Λ_h, Λ_m, and Λ_l, respectively. The packet lengths at the input of the queues (after chopping and selection) are of distributions x_h, x_m, and x_l, where $x_h = X_h$ and $x_m = X_m$. The arrival rates corresponding to x_h, x_m, and x_l are $\lambda_h = \Lambda_h$, $\lambda_m = \Lambda_m$, and λ_l, respectively.

Highest- and Medium-priority Queues. We begin the derivation from the easiest part. Because this scheme is a special case of the head-of-line priority classification scheme and because the arrival processes to the queues are Poisson (except for the lowest-priority queue), we can use the classical work to obtain the waiting times at the highest-priority and medium-priority queues.[5] They can be written as:

$$E[W_h] = \frac{E[W_0]}{(1 - \rho_h)} \tag{A-1}$$

$$E[W_m] = \frac{E[W_0]}{(1 - \rho_h - \rho_m)(1 - \rho_m)} \tag{A-2}$$

where $\rho_h = \lambda_h E[x_h]$ and $\rho_m = \lambda_m E[x_m]$ are the traffic utilizations at the STAT MUX service speed and $E[W_0]$ is the expected residual service time for the one in service. It can be written as:

$$E[W_0] = \frac{\lambda_h E[x_h^2] + \lambda_m E[x_m^2] + \lambda_l E[x_l^2]}{2} \tag{A-3}$$

As is obvious, the waiting time in the queue does not "explicitly" depend on the messages to the lower-priority queues, except for their contribution to $E[W_0]$. Here, ρ_h is the sum of traffic utilizations due to polls from high-priority data links $\rho_{p,m}$ and from low-priority data links $\rho_{p,l}$.

The Low-priority Queue. The difficulty in obtaining a closed-form formula for the lowest-priority packet delay is because its arrival process is non-Poisson and its message length distribution is general. This is a G/G/1 problem. Fortunately, as Morgan and Lo point out, the G/G/1 conservation law provides a way to get around it.[6] As we can see from the following equation, with the knowledge of $E[W_h]$ and $E[W_m]$, the G/G/1 conservation law (a linear equation that relates waiting times) provides a way to solve for the unknown lowest-priority packet delay.[6]

$$\rho_h E[W_h] + \rho_m E[W_m] + \rho_l E[W_l] = E[U] - E[W_0] \tag{A-4}$$

where $E[U]$ is the expected value of the unfinished work in the system. As Morgan and Lo point out, it might appear simpler to apply the equation above directly to the packets at the queues. However, the proof of the G/G/1 law involves the assumption that the arrival of packets is not correlated in any way with the length of the packets. To avoid the possible difficulty in applying the G/G/1 law, a more elaborate approach assuming infinitesimal subpackets

is suggested in Reference 6. The subpacket delay can then be related to the packet delay. Assuming infinitesimal subpackets, the G/G/1 conservation law becomes:

$$\rho_h E[v_h] + \rho_m E[v_m] + \rho_l E[v_l] = E[U] - \rho_h \frac{E[\epsilon_h^2]}{2E[\epsilon_h]} - \rho_m \frac{E[\epsilon_m^2]}{2E[\epsilon_m]} - \rho_l \frac{E[\epsilon_l^2]}{2E[\epsilon_l]} \quad \text{(A-5)}$$

where v_l, v_m, and v_h are the waiting times, and ϵ_l, ϵ_m, and ϵ_h are the lengths of subpackets.

As the subpacket length approaches zero, equation (A-5) becomes:

$$\rho_h E[v_h] + \rho_m E[v_m] + \rho_l E[v_l] = E[U] \quad \text{(A-6)}$$

where $E[U]$ can be obtained from any work-conserving discipline for the M/G/1 message. In particular, the formula for the first-come, first-served discipline is used to obtain $E[U]$ as:

$$E[U] = \frac{\rho E[X^2]}{2(1 - \rho)E[X]} \quad \text{(A-7)}$$

where $\rho = \rho_h + \rho_m + \rho_l$ is the total traffic utilization. Let $F^c(S) = Pr(X>S)$ be the complementary cumulative probability distribution function of X. It can then be decomposed as:

$$F^c(S) = \left(\frac{\Lambda_l}{\Lambda}\right) F_l^c(S) + \left(\frac{\Lambda_m}{\Lambda}\right) F_m^c(S) + \left(\frac{\Lambda_h}{\Lambda}\right) F_h^c(S)$$

in terms of its components, where $\Lambda = \Lambda_l + \Lambda_m + \Lambda_h$. Then,

$$E[X] = \int_0^\infty F^c(S)\, dS$$

$$E[X^2] = 2\int_0^\infty SF^c(S)\, dS$$

If the delay of the first packet in a lowest-priority message is known and if the average interval between serving two consecutive packets of the same message can be found, the entire message delay can be obtained. Consider a message of N packets. Let the expected waiting time of the first packet of a given message be τ, let σ be the interval between serving two packets of a given message, and let δ be the service quantum. The delay of the whole message in the low-priority queue is defined as:

$$E[W_l] = \tau + (N - 1)\sigma \quad \text{(A-8)}$$

where

$$\sigma = \Lambda_l \delta E[X_l] + \Lambda_h(\delta + \sigma)E[X_h] + \Lambda_m(\delta + \sigma)E[X_m] = \rho_l \delta + (\rho_h + \rho_m)(\delta + \sigma) \quad \text{(A-9)}$$

To use the G/G/1 conservation law, however, we need to relate $E[W_l]$ to $E[v_l]$. Note that the number of packets of infinitesimal length ϵ in a message of length X is X/ϵ. Then, the mean waiting time of the subpackets in a high-priority message is

$$V_h(X) = \frac{1}{\epsilon}\int_0^X \left[E[W_h] + z\right] dz = \frac{1}{\epsilon}\left[E[W_h]X + \frac{X^2}{2}\right] \quad \text{(A-10)}$$

Letting $f_h(S)$ be the probability density function of the high-priority message lengths, we obtain $E[v_h]$ by averaging over all high-priority subpackets as:

$$E[v_h] = \frac{\int_0^\infty V_h(S) f_h(S)\, dS}{\frac{1}{\epsilon}\int_0^\infty S f_h(S)\, dS}$$

$$= E[W_h] + \frac{E[X_h^2]}{2E[X_h]} \qquad \text{(A-11)}$$

Similarly, we get

$$E[v_m] = E[W_m] + \frac{E[X_m^2]}{2E[X_m]} \qquad \text{(A-12)}$$

To get $E[v_l]$, we need to calculate $V_l(X)$, the mean waiting time of the subpackets in a low-priority message. Assume that a low-priority message has length X. The number of packets in this message is $N = \lceil X / \delta \rceil$. The time in queue seen by the subpackets in the message is, on the average,

$$V_l(X) = \frac{1}{\epsilon}\int_0^X \left[\tau + \sigma\left(\left\lceil \frac{z}{\delta}\right\rceil - 1\right) + z\right] dz$$

$$= \frac{1}{\epsilon}\left[\tau X + \sigma\int_0^X \left(\left\lceil \frac{z}{\delta}\right\rceil - 1\right) dz + \frac{X^2}{2}\right] \qquad \text{(A-13)}$$

Letting $f_l(S)$ be the probability density function for low priority message lengths, we get $E[v_l]$ as:

$$E[v_l] = \frac{\int_0^\infty V_l(S) f_l(S)\, dS}{\frac{1}{\epsilon}\int_0^\infty S f_l(S)\, dS}$$

$$= \tau + A\sigma + \frac{E[X_l^2]}{2E[X_l]} \qquad \text{(A-14)}$$

where

$$A = \frac{1}{E[X_l]}\int_0^\infty \int_0^S \left[\left\lceil \frac{z}{\delta}\right\rceil - 1\right] dz\, f_l(S)\, dS \qquad \text{(A-15)}$$

By changing the order of integration, we get:

$$A = \frac{1}{E[X_l]}\int_0^\infty \int_z^\infty \left[\left\lceil \frac{z}{\delta}\right\rceil - 1\right] f_l(S)\, dS\, dz$$

$$= \frac{1}{E[X_l]}\int_0^\infty \left[\left\lceil \frac{z}{\delta}\right\rceil - 1\right] F_l^c(S)\, dS$$

$$= \frac{1}{E[X_l]}\sum_{N=0}^\infty N\int_0^\delta F_l^c(Z + N\delta)\, dZ \qquad \text{(A-16)}$$

Finally, we can solve for τ by using equations (A-6), (A-7), (A-11), (A-12), and (A-14). The low-priority message delay can then be obtained by calculating (A-8).

Appendix B. Modifying Walk-time Equations and End-to-End Delay Formulas

If polls are given priority at the STAT MUX, the delay that a poll encounters at STAT MUX is:

$$E[W_h] = \frac{E[W_0]}{(1-\rho_h)}$$

as shown in equation (A-1). This extra delay should be added to each walk time w_i discussed in the section on the virtual-circuit data network. The reader should notice that we have two sets of end-to-end delay formulas and walk time equations, one for each class of applications.

In the case that priority at the STAT MUX is granted based on application, messages and polls from priority applications are put in a high-priority queue and others are partitioned, intermingled, and put in a low-priority queue. This is a two-queue approach, which is very close to the priority scheme used in ISN, except that priority is granted based on application rather than message size. The traffic utilization due to the high-priority messages ρ'_h is $\rho_m + \rho_{p,m}$, where $\rho_{p,m}$ denotes the traffic utilization at the high-priority queue output due to the poll frames of high-priority applications. The traffic utilization due to low-priority messages ρ'_l is $\rho_l + \rho_{p,l}$, where $\rho_{p,l}$ denotes the traffic utilization at the low-priority queue output due to poll frames of low-priority applications. The extra delay of messages and polls from high-priority data links can be obtained using the same approach in Appendix A as follows:

$$\frac{E[W_0]}{(1-\rho'_h)}$$

This component should be added to the walk time equations and the end-to-end delay formula of high-priority data links. The extra walk time for a poll from low-priority data links is τ', the counterpart of τ in Appendix A. Following the same approach, τ' can be easily derived. The extra message delay, $\tau' + (N-1)\,\sigma$ should be added to the end-to-end delay formula of low-priority data links.

If the messages are placed into a single queue and the service quantum is a message, the delay for any poll can be obtained using the M/G/1 formula as:

$$\frac{\rho E[X^2]}{2E[X](1-\rho)}$$

where $\sigma' = \rho'_l\,\delta + \rho'_h(\sigma + \delta)$.

Biographies (continued)

signal processing and coding for sources, channels, and networks. He has a B.S. in electrical engineering from Middle East Technical University, Ankara, Turkey, and an M.S. and a Ph.D. in electrical engineering from Stanford University.

(Manuscript received April 25, 1988)