Reliable Link Initialization Procedures

ALAN E. BARATZ AND ADRIAN SEGALL, FELLOW, IEEE

Abstract—**It is known that HDLC and other bit-oriented DLC procedures ensure data transmission reliability across noisy transmission media provided that all frame errors are detected and the link processes are synchronized at initialization. We show in this paper that the HDLC initialization procedure does not ensure synchronization and allows inadvertent loss of data. We then propose several new link initialization procedures and prove that they do ensure synchronization.**

I. Introduction

DATA link control (DLC) procedures are protocols that use error detection and retransmission mechanisms to protect data sent over a noisy transmission media from transmission errors. The main role of such procedures is to accept data at one end of a link and ensure their delivery at the other end in the same order as accepted, without losses or duplicates. As long as the link works properly, all data should arrive at the receiving end in finite time. If data accepted by a DLC procedure cannot be transmitted because of link failure, appropriate notification should be submitted to the higher layers. DLC procedures which guarantee these properties are said to provide *data reliability*.

The most commonly used DLC procedures are the bit-oriented DLC procedures such as HDLC [3], SDLC [4], ADCCP [1], or alternating bit [2]. In these procedures, there are three main situations that may result in undetected transmission errors: 1) improper design of the DLC procedure, 2) undetected frame errors, and 3) incorrect initialization of the DLC procedure.

There is no way to ensure that undetected transmission errors will never occur under any circumstance. There is always the possibility that a logically correct program will execute improperly due to hardware or system errors. Moreover, any error detection scheme has an inherent probability of undetected errors. Consequently, all work on reliable DLC procedures is directed towards minimizing the probability of undetected transmission errors.

The first two issues above have received great attention in the literature. Data reliability is ensured by HDLC and the other bit-oriented DLC procedures as long as the DLC processes are synchronized at initialization and all frame errors are detected. Moreover, frame error detection has been well studied and powerful cyclic redundancy checking schemes are currently used to minimize the probability of undetected errors. The main subject of this paper is to introduce link initialization (LI) procedures for which the possibility of incorrect initialization is significantly lower than that for presently known initialization procedures.

Paper approved by the Editor for Network Protocols of the IEEE Communications Society. Manuscript received June 19, 1986; revised April 20, 1987. A preliminary version of this paper was presented at the 3rd International Workshop on Protocol Specification, Testing, and Verification, Zurich, Switzerland, May 1983.

A. E. Baratz is with the IBM T. J. Watson Research Center, Yorktown Heights, NY 10598.

A. Segall was on leave at the IBM T. J. Watson Research Center, Yorktown Heights, NY 10598. He is with the Department of Electrical Engineering, Technion—Israel Institute of Technology, Haifa 32000, Israel.

IEEE Log Number 8718650.

The best currently known LI procedures is the one used by HDLC [1], [3], [7], [10]. However, in Section III, we demonstrate scenarios where the HDLC LI procedures fail to provide synchronization and allow inadvertent loss of data as a result of nothing more than unpredictably long link delays (e.g., due to long queueing caused by congestion); these scenarios assume no loss of memory or failure of any other type at the link stations. Moreover, it has been conjectured [8] that even if the link stations work correctly, "if transmission delays between processes are variable and finite, but no upper bound is known *a priori,* then absolutely reliable techniques for closing and opening a connection (i.e., LI procedures) do not exist." In contrast, our new LI procedures completely eliminate such situations and ensure synchronization for all cases of link malfunction, as long as there is no loss of memory in the link station.

Our LI procedures possess two additional appealing features. First, their implementation complexity is not greater than that of the HDLC LI procedure. Second, they can be adapted to cope with link station failures, provided only two bits of nonvolatile memory exist in each of the stations (or only in the primary station for the unbalanced mode). As mentioned before, no procedure is completely failsafe against all types of failures. In particular, if the nonvolatile memory fails, then loss of synchronization and inadvertent loss of data may occur, even with our new LI procedures. However, we have considerably reduced the possibility of error as compared to existing LI procedures.

In Section II, we formalize the notions of a link, a bit-oriented DLC procedure, and an LI procedure. In Section III, we show that the LI procedure used by HDLC does not ensure synchronization and may lead to inadvertent loss of data with no notification to the higher layers. Finally, in Section IV, we present the new LI procedures.

II. The Model

In this section, we formalize the notions of a link, a bit-oriented data link control procedure, and a link initialization procedure. We also introduce some terminology that will be used throughout the remainder of the paper.

A. Basic Components

The configuration considered in this paper is given in Fig. 1. *Data source* is a generic name for some device, process, or higher layer that produces data strings which have to be transmitted over a communication link to a *data sink.* We shall assume that the data strings are packetized and shall refer to them as *packets.* Furthermore, we assume that the packets are queued in a buffer at the data source and that they are transferred *in order* to the DLC process *at times dictated by the* DLC *procedure.* At the other end, the packets are delivered by the DLC process to the data sink at times dictated by the DLC procedure.

A bit-oriented *data link control procedure* is a pair of processes, one at each station, that operate together using some type of acknowledgment and retransmission scheme to ensure data reliability over an error-prone transmission media. A DLC process accepts packets from the data source, transforms them into *information frames* by appending any

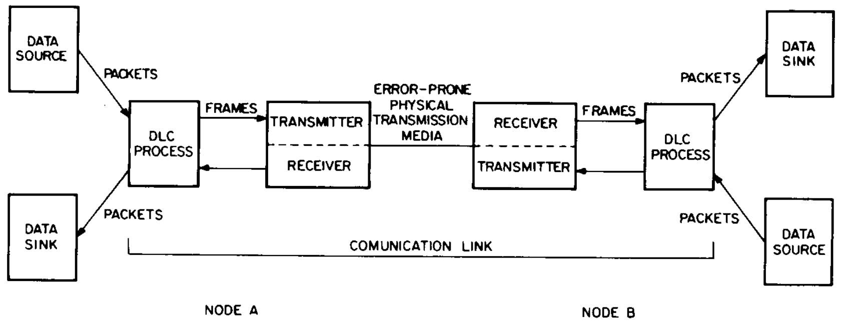


Fig. 1. The communication system.

necessary control or sequencing information, and transfers them to the transmitter. In addition, it accepts incoming frames from the receiver, converts them back into packets, and sends them on to the data sink. A more precise description of the DLC procedure will be given in Section II-B.

The DLC processes are served by a point-to-point communication connection between two communicating stations. The transmitter at one station receives the information frames and transforms them into transmission *units* by appending to each an error checking code. The units are then queued for transmission in the same order as received. Units sent by the transmitter may be lost, arrive at the receiver in error, (e.g., because of transmission noise), or arrive correctly after some finite but unknown delay. We assume that all errors are detected by the receiver and the units received in error are discarded. Units that arrive correctly are assumed to arrive in the same order as sent and are transferred, after deletion of the error checking code, to the corresponding DLC process. Thus, we assume that the transmission media preserves FIFO ordering. However, we do not require that a bound on the transmission delay is known *a priori*. Finally, we assume that when the physical transmission media is operational, the probability that a unit is lost or received in error is less than 1.

The entire system shown in Fig. 1, except for the data sources and sinks, will be referred to as the *communication link*. This includes the DLC processes and their storage, the transmitters, the receivers, and the physical transmission media.

The procedures used by the DLC processes to transmit data reliability over the error-prone transmission media are referred to as DLC procedures. Each such procedure is composed of two stages: initialization stage and connected stage. The purpose of the initialization stage is to synchronize the two DLC processes and clean the channel of old information, while in the connected stage, the processes exchange information data.

In the present paper, we are mainly concerned with the initialization stage of the DLC procedure, although some discussion about the connected stage will also be necessary. Existing procedures implement the initialization stage by using special control messages, acknowledges, and time outs. A more detailed description will be presented in Section III, but here we point out that one of the critical requirements of those existing procedures is that the time-out period is larger than the maximal roundtrip delay of the control messages. This poses a major problem in the design of the time-out periods. As shown in the examples of Section III, time outs that are too short may result in nonreliable communication and inadvertent loss of frames. On the other hand, time outs that are too long result in a long setup time because the DLC process waits longer than necessary before retransmitting control messages that are lost. The time-out interval in the existing procedures has to be set to a large enough value to ensure that the link is free of old control messages before it expires, but such a value may be much larger than it is normally necessary. Moreover, DLC procedures are often used on transmission media like satellite with on-board processing, gateways, local area networks, etc., for which if is hard to determine *a priori* bounds on the transmission delay. For such environments, it is difficult to establish a time-out interval that is tight on one hand and not exceeded by the round-trip delay on the other hand.

The main purpose of this paper is to propose new initialization procedures that work without the stringent requirement that the time-out interval exceeds the roundtrip delay. Their main property is that DLC process synchronization and channel clearance are achieved under arbitrary channel delays. Another feature of the proposed procedure is that their complexity is similar to the one of the existing procedures, so that the increased reliability is not achieved at the expense of increased complexity.

B. The DLC Procedure

In this section, we define a *general class* of bit-oriented DLC procedures to which we shall refer as *window-DLC procedures*. Our description will focus on the interaction required between the two DLC processes to transmit data from one station (say A) to the other station (say B). To facilitate our discussion, the DLC process at station A will be referred to as the *sender* DLC and the DLC process as station B will be referred to as the *receiver* DLC. Note that, in general, each DLC process will be acting as both sender and receiver since data will be transmitted in both directions. Except for some minor notational differences, all known bit-oriented DLC procedures (e.g., alternating bit, HDLC, SDLC, etc.) are members of the class of window-DLC procedures.

In a window-DLC procedure, the sender DLC maintains a *send counter number*, denoted[1] by VS, and the receiver DLC maintains a *receive counter number*, denoted by VR. The range of VS and VR is between 0 and $W - 1$ where W is some fixed integer (the quantity $(W - 1)$ is called the *window size*; the alternating bit protocol used $W = 2$ and the HDLC protocol uses $W = 8$ or $W = 128$). Initialization of the counter numbers is performed by the link initialization procedure and will be discussed later. For now, it suffices to say that the relation $VS = VR$ must hold at initialization.

Once the send counter number is initialized, the sender DLC is allowed to accept $W - 1$ packets from the data source. These packets are assigned consecutive *sequence numbers* from VS to $(VS + W - 2)$ modulo W. The sender DLC transforms each accepted packet into an information frame by appending a control header containing the assigned

[1] To avoid confusion, we point out that the quantity VS here is different from $V(S)$ of HDLC [3]. However, for our purpose, it is more convenient to use this notation.

sequence number (NS). An information frame is stored by the sender DLC from the time the corresponding packet is accepted until it is considered acknowledged, at which time it is discarded. We leave unspecified and arbitrary the time when the sender DLC may send copies of the stored frames. Various DLC procedures dictate the instances when frames (and their copies) are actually sent, but this is of no concern to us here.

When a frame with sequence number $NS = VR$ is received correctly at the receiver DLC, it is delivered (without the control header) as a packet to the data sink and VR is incremented by 1 modulo W. Frames received in error or with sequence number $NS \neq VR$ are discarded and no action is taken. The receiver DLC also has some mechanism to periodically send an *information ACK frame* containing *acknowledgment number* $NR = VR$ to the sender DLC. Whenever an information ACK frame with acknowledgment number $NR \neq VS$ arrives at the sender DLC, VS is repeatedly incremented modulo W until it reaches NR. Note that VS is not incremented when frames are sent out, but rather only when an appropriate ACK frame is received. In addition, when VS is incremented from value K to $(K + 1)$ mod W, the stored frame that carries sequence number K is *considered acknowledged* and discarded. Then a new packet is accepted from the data source and is assigned sequence number $(K - 1)$ mod W. Observe that at any time the sender DLC stores at most $W - 1$ frames, with sequence numbers from VS to $(VS + W - 2)$ modulo W. As said before, we do not specify here the times when the sender DLC is allowed to send its stored frames or when the receiver DLC sends the acknowledgment. However, the sender DLC process is required to periodically send out the information frame with sequence number $NS = VS$. Similarly, the receiver DLC process is required to send out periodically an acknowledgment. In other words, the stations are not allowed to be idle (in the sense of not performing the DLC procedure) indefinitely.

We now note that different window-DLC implementations employ various additional optimization techniques to achieve efficient link utilization. Examples include appending the ACK frame to information frames being sent in the opposite direction, using NACK frames (selective reject) and/or checkpointing and saving information frames with $NS \neq VR$ for later use while reducing the corresponding windows. Although important from the performance point of view, these techniques need not be discussed here. The properties of the proposed initialization procedures will be shown to hold for any window-DLC procedure.

In addition to *normal operation* (as described above), DLC procedures must include mechanisms for detecting link failures, mechanisms for detecting link recoveries, and a reinitialization protocol that will allow the resumption of normal operation. The failure detection mechanism must have the basic property that *when the* DLC *process is in normal operation, either every frame sent by the process is considered acknowledged within finite time or a link failure is detected.* The most common failure detection mechanisms are to declare a link as failed if a frame is not acknowledged after a given number of retransmissions or after a predetermined time. Whenever a DLC process detects a failure, it declares any unacknowledged information frames as possibly lost and forwards appropriate notification to the higher layers. It then invokes a reinitialization protocol that first probes the channel periodically for detection of link recovery and then resynchronizes the system for resumption of normal operation. When the reinitialization protocol terminates, the DLC process resumes normal operation. In this paper, we shall say that a DLC process is in *connected state* when it performs normal operation and that it is in *initialization mode* otherwise (i.e., when it is executing the initialization procedure). The mechanism for detecting link failures is active at a node during the time the DLC process is in connected state, but it is inhibited when the DLC process is in initialization mode.

The following definition formalizes the notion of reliability for a DLC procedure.

Definition: A bit-oriented DLC procedure is said to ensure *data reliability* if it satisfies *all* of the following properties.

1) Delivery: Whenever a DLC process is in connected state, all packets that have been produced by its data source since it last entered connected state are delivered to the corresponding data sink within finite time, provided the DLC process does not enter initialization mode.

2) FIFO: Whenever a DLC is in connected state, all packets accepted from its data source since it last entered the connected state and considered acknowledged have been delivered to the corresponding data sink and with no losses or duplicates.

3) Crossing: If a DLC process (say A) enters initialization mode at some time t_1, there is a time t after t_1, but before A next enters connected state, such that DLC process B is also in initialization mode and no packet accepted by the sender DLC at either end before time t can be delivered to the corresponding data sink after time t.

4) Deadlock-Free: There exists a value T_1 such that if a) both DLC processes are in initialization mode at some time t and b) during the interval of length T_1 after t there are no channel errors and c) the delay for all frames (queueing + propagation) is bounded, then at time $t + T_1$ both processes are in connected state.

The *Delivery* property states that if the DLC processes stay in connected state forever, packets produced by data sources eventually get delivered to the data sinks. *FIFO* states that those packets that are considered acknowledged indeed have been delivered correctly and in sequence. *Crossing* relaxes and formalizes the usual notion of a "correct global initial state" (see, e.g., [13]) where both processes are in connected state with sequence numbers = 0 and the channel is empty. However, it may be the case that one DLC process enters connected state and starts sending frames before the other enters connected state, so that strictly speaking, there is no instant when the system is in a correct global initial state. In this situation, we still think of the DLC procedure as reliable, provided it satisfies the property indicated above under *Crossing*. Finally, the *Deadlock-Free* property says that if the channel works properly, the processes are not deadlocked in initialization mode.

Observe that data reliability ensures the proper delivery of packets corresponding to frames that are considered acknowledged, but gives no indication as to the fate of unacknowledged packets (there can be at most $W - 1$ of them). In particular, if a DLC process enters initialization mode it should notify the higher layers that these packets have not been acknowledged and, consequently may have been lost.

In Section II-C, we will show that any window-DLC procedure ensures data reliability if properly synchronized at initialization. However, we must first formalize the notions of synchronization and of a link initialization procedure.

C. The LI Procedure

The procedure that allows DLC processes to resynchronize after a link failure is the main subject of this paper and will be referred to as the *link initialization (LI) procedure.* An LI procedure consists of a pair of processes, one residing in each station, that communicate with each other through special *LI*-control frames (the equivalent of a subset of the unnumbered frames in HDLC [1], [3]). After appropriate frame exchange, the LI procedure should bring both DLC processes into connected state. The following definition formalizes the notion of synchronization for an LI procedure.

Definition: An LI procedure working in conjunction with a bit-oriented DLC procedure is said to ensure *synchronization* if it satisfies the following properties.

1) Clear: If a DLC process (say A) enters initialization mode at some time t_1, there is a time t after t_1, but before A next enters connected state, such that DLC process B is also in initialization mode and the communication link is devoid of any information frames and any information ACK frames.

2) Reset: If a DLC process (say A) enters connected state while the other DLC process (B) is already in connected state, then the send (VS) and receive counter numbers at A are identical to the receive (VR) and send counter numbers, respectively, at B.

3) Deadlock-Free: There exists a value T_1 such that if a) both DLC processes are in initialization mode at some time t and b) during the interval of length T_1 after t there are no channel errors, and c) the delay for all frames (queueing + propagation) is bounded, then at time $t + T_1$ both processes are in connected state.

The following theorem demonstrates the relationship among a bit-oriented DLC procedure, its associated LI procedure, and the notion of data reliability.

Theorem 2.1: If a window-DLC procedure has an LI procedure which ensures synchronization, then the DLC procedure ensures data reliability.

The above theorem has been proved for HDLC in several previous works, e.g., [5], [13]. Here we prove reliability for any window-DLC. The proof appears in the Appendix.

In Section III, we will describe the LI procedure used by HDLC and show that it does not ensure synchronization in cases of link malfunction or unpredictable message delay. In fact, we will show that HDLC link initialization allows inadvertent loss of information frames and thus does not ensure reliability. In Section IV, we will describe three new LI procedures that do ensure synchronization, even if an upper bound on the transmission delay is not known *a priori*. The first of our procedures assumes that one of the stations is predesignated as the primary station and the other as the secondary. The second procedure assumes no such preassignment, but begins by explicitly assigning primary/secondary functions. In the third procedure, no preassignment is assumed and no postassignment is performed. In addition to ensuring synchronization, these three LI procedures also satisfy one additional property that may be important in some environments.

Test: A DLC process in initialization mode should not enter connected state before it observes that the roundtrip delay across the communication link is within a prespecified bound for at least one frame.

The purpose of the test property is to provide insurance that the link stations will not unnecessarily oscillate between the connected state and initialization mode.

D. Notation

In the remainder of this paper, we shall use finite state diagrams (see Fig. 2) to describe the operation of various LI procedures. The notation T/A next to a state transition will denote the fact that T is the trigger for that transition and A is the action to be taken *after* transition. The triggers will generally be the receipt of particular LI-control frames or a time out. We shall adopt the convention that receipt of any frame other than those specified as triggers causes no action (i.e., the message is disregarded and discarded). The term *reset* will denote the resetting of all counter numbers and the clearing of all memory associated with the communication link at the given station.

In order to describe the sequence of events that may occur across a link, we will use a timing diagram with two parallel time axes (one for each link station) as shown in Fig. 3. An arrow drawn from one axis to the other represents a frame sent by one station and received by the other. An arrow that does not terminate on a time axis represents a lost frame (e.g., a frame that arrives in error and is discarded). The tail of each arrow is labeled with the corresponding message type. LI-control frames are represented by solid arrows and information frames or information ACK frames are represented by dashed arrows. I_n represents an information frame with send sequence number n and ACK_m represents an ACK frame with acknowledge sequence number m.

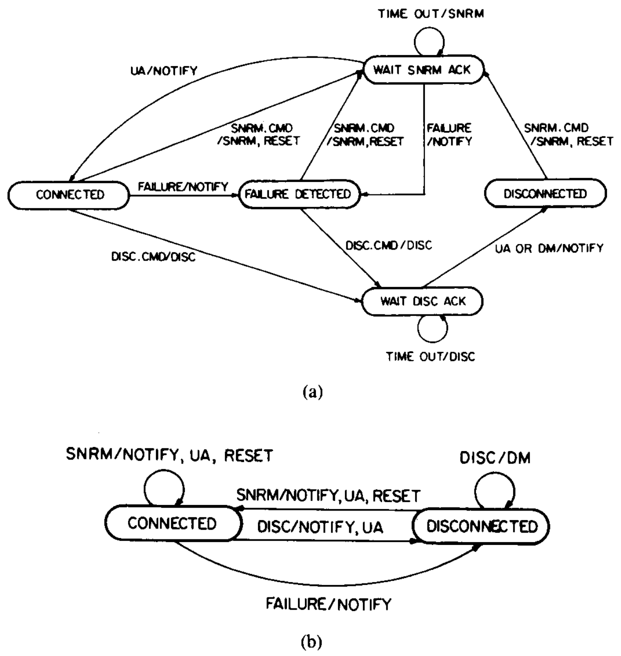


Fig. 2. The HDLC link initialization procedure. (a) Primary station. (b) Secondary station.

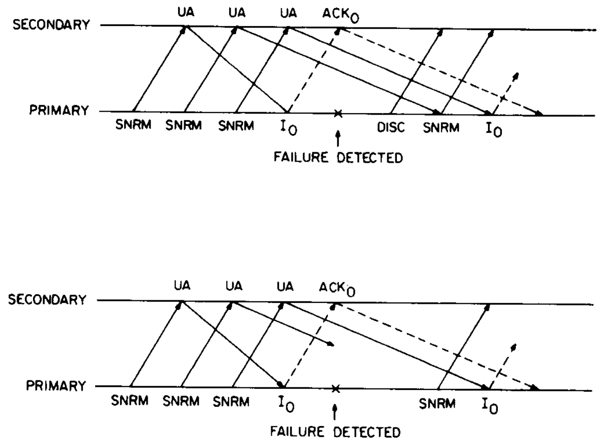


Fig. 3. Examples of inadvertent loss of frames.

III. The HDLC Link Initialization Procedure

In this section, we describe the link initialization procedure used by HDLC in the unbalanced normal response mode and show that is does not ensure synchronization. The finite state diagrams describing the HDLC link initialization procedure are shown in Fig. 2 [7]. (Fig. 2 is the same as [7, sect. 5], except it also shows time outs and exchange of messages with the higher layer). Set normal response mode (SNRM), disconnect (DISC), unnumbered acknowledgment (UA), and disconnect mode (DM) are the LI-control frames. NOTIFY means "notify the higher layers" and RESET means "reset sequence number." Note that the actual operation of this LI procedure is dependent on information obtained from a higher layer, allowing for various versions of operation. In particular, transition from the failure detected and disconnected states is triggered by the receipt of instructions SNRM.CMD or DISC.CMD from a higher layer. However, we will show that all versions of this LI procedure do not ensure synchronization, independent of the action taken by the higher layers.

Consider the possible sequence of message exchanges shown in Fig. 3(a).[2] This sequence begins with the primary station entering the Wait SNRM Ack state and the secondary station in the disconnected state, a common configuration. After sending an SNRM message, the primary times out. When the timer expires, another SNRM is sent. Normally, the timer is set such that if a UA is not received within its range, there is a good chance that the SNRM or the UA has been lost. However, as indicated in Section II-A, in some situations it may be hard or inefficient to guarantee that this is always the case. The situation considered in Fig. 3 is where the timer expires twice before the UA for the first message is received.

Following detection of a link failure, the primary may send either an SNRM or a DISC frame, depending on the instructions received from the higher layer. Fig. 3(a) demonstrates the situation where the primary sends a DISC frame. This is followed by the receipt at the primary of an ACK_0 frame that is interpreted to acknowledge an information frame that, in fact, has been lost (the second I_0 frame). Fig. 3(b) shows that the same problem may result when the primary sends an SNRM frame after the failure detection. Notice that two of the properties required to ensure synchronization (clear and reset) are violated by this LI procedure no matter what action is taken by the higher layers. Thus, the HDLC link initialization procedure does not ensure synchronization and, moreover, HDLC itself does not ensure data reliability.

IV. Reliable Link Initialization Procedures

One way to prevent the situation described in Section III and correct the HDLC link initialization procedure is to use sequence numbers. The LI-control frames could carry correlated sequence numbers. However, infinitely large sequence numbers would be required to ensure synchronization. In this section, we describe several new LI procedures that ensure synchronization without the use of sequence numbers or time stamps. In Section IV-A, we address the situation where one of the link stations is predesignated as the primary and the other as the secondary. In Section IV-B, we consider the case where there is no such preassignment, but the LI procedure begins by explicitly assigning primary and secondary functions. In Section IV-C, we describe an LI procedure where no primary/secondary preassignment is assumed and no postassignment is performed (i.e., a fully balanced procedure). The last two procedures have the property that the finite state machines in each link station are identical.

A. An Unbalanced LI Procedure

In this section, we address the situation where one of the link stations is predesignated as the primary and the other as the secondary. The finite state diagrams describing the algorithm performed by each station are given in Fig. 4. The primary station begins by transmitting DISC control frames at regular time intervals until receiving a DACK from the secondary. It then transmits CLEAR control frames at regular time intervals until receiving a CACK. Finally, it transmits exactly one TEST frame and waits for a TACK. If the TACK arrives within the specified time period, the primary transmits

[2] All unnumbered frames considered here have the *P/F* bit set to 1 (see [3, pp. 23–25] for details).

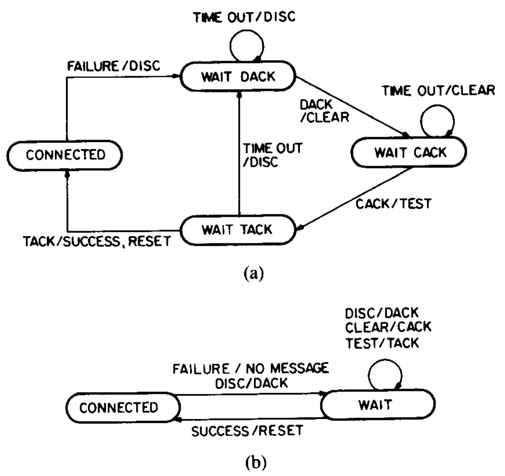


Fig. 4. The new unbalanced LI procedure. (a) Primary station. (b) Secondary station.

a SUCCESS frame, resets its send and receive counter numbers, and enters the connected state. Otherwise, the entire process is repeated beginning with the DISC frame. The secondary station simply transmits a DACK, CACK, or TACK frame whenever it receives a DISC, CLEAR, or TEST, respectively. In addition, when the secondary receives a SUCCESS frame, it resets its send and receive counter numbers and then enters the connected state. A link station in initialization mode ignores all information and information ACK frames and inhibits the link failure detection mechanism. Similarly, a link station in the connected state ignores all LI-control frames other than DISC.

The LI procedure uses three types of TIME-OUT intervals: between DISC messages, between CLEAR messages, and after sending the TEST message. The DISC and CLEAR time outs are arbitrary from a correctness point of view. The LI procedure ensures synchronization for any values of these time outs. The only reason not to make them too short is performance, since this will unnecessarily clutter the channel with DISC and CLEAR messages. The purpose of the wait TACK state and the associated time out is to provide a certain degree of insurance that the channel works properly before returning to connected state. Without the wait TACK state, the LI procedure would still ensure synchronization, but it may bring the link up (to connected state) even if the link is very bad. In the form given in Fig. 4, we are sure that the *Test* property is satisfied, namely, at least one roundtrip delay across the communication link did not exceed the bound specified by the TEST time out. Therefore, the TEST timeout should be set to comply with the required specification. In fact, in various implementations, the wait TACK state and the corresponding TIME OUT may be disposed of if no *test* property is needed or may be replaced by another "box" if there is a different *Test* requirement.

The procedure of Fig. 4 will be referred to as the *unbalanced LI procedure* and the remainder of this section will be devoted to proving its correctness. We will first show that the procedure, as stated, ensures synchronization and satisfies the *Test* property as long as the primary link station does not fail execution of the LI procedure. We will then show how to extend the protocol to handle environments in which the primary station can fail during execution of the LI procedure. This extension will require one bit of nonvolatile memory at the primary.

In order to prove the correctness of the unbalanced LI procedure, we first observe that every TACK frame received by the primary must have been transmitted in response to some

TEST frame by the secondary. Two messages related in this way will be called a *TEST–TACK pair.* Note that whenever the TEST frame in a particular TEST–TACK pair is processed by the primary before the timer expires, the message pair is said to constitute a *successful TEST–TACK transmission* (see Fig. 5). The notion of a successful DISC–DACK transmission and a successful CLEAR–CACK transmission can be similarly defined. We shall use the term *successful roundtrip transmission* as a general reference to any one of these three types of successful transmission.

We now present a technical lemma which describes the unbalanced LI procedure's key operating property. We assume that the primary link station does not fail during execution of the LI procedure, but that the physical transmission media and the secondary link station may fail at any time. When the primary beings operating, it enters the wait DACK state. Every time *the secondary comes up after a failure,* it enters the WAIT state.

Lemma 4.1: If the primary link station enters initialization mode at a time when the communication link contains no DISC or DACK frames, then when the primary *next* enters the connected state it will be due to the receipt of a TACK frame which belongs to a successful TEST–TACK transmission.

Proof: Since the secondary station enters WAIT state after it recovers from a station failure, any failure of the secondary link station will have no effect on the operation of the LI procedure other than increased message delay or loss. Similarly, failures of the physical transmission media is equivalent to message loss on the media. Thus, we need only consider the case where there are no media or station failures, but there may be arbitrary message losses and arbitrary long message delays. Now if there are no DISC or DACK frames in the communication link when the primary station enters initialization mode, then due to the FIFO property of the transmission media, there can be only DISC or DACK frames in the communication link when the primary first exists the wait DACK state (see Fig. 6). In particular, there can be no CLEAR or CACK frames in the link at this time. Thus, when the primary next exists the wait CACK state, there can be only CLEAR or CACK frames in the link. This implies that there can be no TEST, TACK, DISC, or DACK frames in the communication link when the wait CACK state is exited. Therefore, if the primary receives a TACK frame while in the wait TACK state, it must belong to a successful TEST–TACK transmission. On the other hand, if the wait TACK timer expires before a TACK frame is received, then there will be no DISC or DACK frames in the communication link when the wait TACK state is exited and the wait DACK state is reentered. It now follows inductively that whenever the primary *next* receives a TACK frame while in the wait TACK state, it must belong to a successful TEST–TACK transmission. Thus, as long as the primary continues operating, it will next enter the connected state as the result of processing a TACK frame which belongs to a successful TEST–TACK transmission.

Corollary 4.1: If the primary link station begins operating at a time when the communication link contains no DISC or DACK frames, then every time the primary completes execution of the unbalanced LI procedure (and enters the connected state), it will be due to the processing of a TACK frame which belongs to a successful TEST–TACK transmission.

Proof: If the primary link station begins operating at a time when the communication link contains no DISC or DACK frames, then by Lemma 4.1 the primary will complete its first execution of the unbalanced LI procedure (and enter the connected state) due to the processing of a TACK frame which belongs to a successful TEST–TACK transmission. Moreover, at such time there will be no DISC or DACK frames in the communication link. Combining this with the

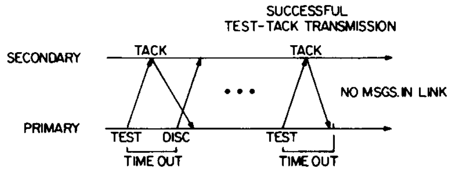


Fig. 5. A successful TEST–TACK transmission.

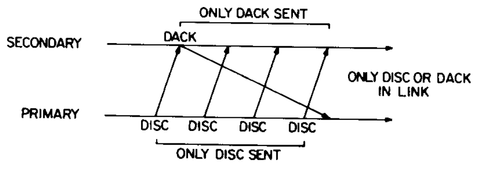


Fig. 6. Timing diagram for proof of Lemma 4.1.

fact that the primary does not send any LI-control frames while in the connected state, it follows that there will be no DISC or DACK frames in the communication link when the primary next begins execution of the LI procedure. Thus, Lemma 4.1 can once again be applied to obtain the same result for the primary's second execution of the LI procedure. It now follows inductively that every time the primary completes execution of the unbalanced LI procedure (and enters the connected state), it will be due to the processing of a TACK frame which belongs to a successful TEST–TACK transmission.

Before proceeding, we should note that Lemma 4.1 addresses only one execution of the LI procedure, while Corollary 4.1 addresses all executions which occur after the primary station is brought up. In addition, we note that Lemma 4.1 does not hold if the primary link station can fail while in initialization mode. The problem arises from the fact that the primary might be restarted in a state other than the one it was in when it failed. This issue will be addressed at the end of this section. For the following theorem, we continue to assume that the primary does not fail while in initialization mode.

Theorem 4.1: The unbalanced LI procedure ensures synchronization and satisfies the test property if the primary link station begins operating at a time when the communication link contains no DISC or DACK frames.

Proof: If the primary link station begins operating at a time when the communication link contains no DISC or DACK frames, then from Corollary 4.1, it follows that every time the primary completes execution of the LI procedure (and enters the connected state), it must be due to the processing of a TACK frame which belongs to a successful TEST–TACK transmission. Thus, whenever the primary enters the connected state, the secondary will be in the WAIT state and there will be no information frames or information ACK frames in the communication link. We now note that the secondary can enter the connected state only after receiving a SUCCESS frame. Combining this with the fact that the primary sends a SUCCESS frame only when it completes execution of the LI procedure, we conclude that the unbalanced LI procedure satisfies the clear property. To see that the reset property is satisfied observe, in addition, that a link station's send and receive counter numbers are reset to zero every time it enters the connected state. Therefore, the primary sets *VS* and *VR* to zero when it sends the SUCCESS message. These sequence numbers remain unchanged until the time the secondary receives the SUCCESS message and enters connected state, in spite of the fact that in the interim the primary may start sending out information frames. This is because *VS* or *VR* can change only when an information ACK frame or an information frame, respectively, is received, and the secondary does not send out any such frames until it enters connected state. At that time, the secondary also resets its counter numbers to zero, and the reset property is satisfied. We now note that once either link station enters initialization mode, return to connected state will not occur until there is at least one successful roundtrip transmission; hence, we have the test property. Finally, if both link stations are in initialization mode and the link becomes capable of performing at least four successive successful roundtrip transmission, then both stations will return to connected state. Thus, the unbalanced LI procedure has the deadlock-free property, completing the proof of the theorem.

Our discussion thus far has been based on the assumption that the primary links station does not fail during execution of the LI procedure. If we drop this assumption, there is the question of what state does the primary station restart in. It is easy to show that no matter which state is selected for restart, Lemma 4.1 no longer holds. Thus, the unbalanced LI procedure must be modified so that its operation is unaffected by primary station failures. This can be easily accomplished if the primary station maintains one bit of nonvolatile memory which is set to 1 every time the wait CACK state is entered, and reset to 0 every time the wait CACK state is exited. Whenever the primary stations fails, it restarts in the wait CACK state if the value of the bit is 1 and in the wait DACK state otherwise. This modification will essentially leave the LI procedure unaffected by primary station failures. This is because if the station fails in wait DACK or wait CACK state, it will come up in the same state. If it fails in connected or wait TACK state, it will come up in wait DACK state, and this transition is equivalent to FAILURE detected on the link or TEST TIME-OUT expiration. Therefore, Lemma 4.1 and Theorem 4.1 will once again apply.

B. A Function-Assigning LI Procedure

In the unbalanced LI procedure of the previous section, it was assumed that the primary/secondary functions were preassigned. This section considers the case where no such preassignment exists, but the LI procedure is required to assign primary/secondary functions to the stations. A procedure of this type is necessary when the stations must use an unbalanced operation mode in connected state (as in SDLC for example). We note, however, that such an LI procedure can also be used if the stations employ a balanced operation mode in connected state by simply disregarding the assignment. The advantage over the balanced LI procedure (to be presented in the next section) is that fewer LI-control messages are used. On the other hand, the balanced LI procedure must be used if there is no ordering of the station identifiers or if there is an explicit requirement that the activity of the LI procedure be completely symmetric.

The function-assigning LI procedure uses a fairly simple rule for primary/secondary assignment. The station that enters first initialization mode becomes the primary, except for the case when each of the two stations enters initialization mode before finding out that the other station has entered this mode as well. In this latter situation, the station with higher identity number becomes the primary. The finite state diagram for station A is shown in Fig. 7. Station B has the same diagram except that A and B are interchanged. The idea is that a station (say A) which detects a link failure while in connected state enters wait DACK state and attempts to become the primary by sending DISC messages to the other station. If it receives a DISC message while in this state, it recognizes that station B has done the same, and if $B > A$, it yields the primary function to the other station by sending DACK and entering WAIT state. Otherwise, station A will receive DACK and declare itself the primary.

The correctness of the function-assigning LI procedure

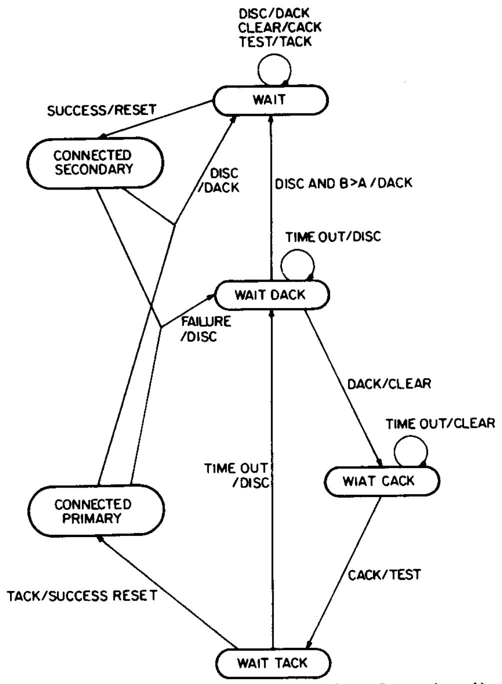


Fig. 7. The function-assigning LI procedure (for station A).

follows directly from the proofs for the unbalanced LI procedure and several additional observations. Consider first the case where the stations do not fail and whenever a station begins operating it enters the wait DACK state. Then note the following.

- A station (say B) can enter WAIT state only if it receives a DISC frame and this can be sent only if the other station is in wait DACK state; WAIT state can be left only by entering connected secondary state.
- A station (say A) can make the transition to the wait CACK state only of it receives a DACK message and this can be sent only if the other station is in WAIT state.

These observations imply that after any station (say A) enters wait CACK state (and until both stations enter connected state), the stations operate exactly as in the unbalanced LI procedure with A the primary and B the secondary! Therefore, all properties of Section IV-A hold here as well.

If the stations may fail while in initialization mode, each needs two bits of nonvolatile memory to indicate if the failure occurs in WAIT state, in wait CACK state, or elsewhere. The station will then restart in WAIT state, wait CACK state, or wait DACK state, respectively. The argument why this leaves the LI procedure unaffected by station failures is the same as in the last paragraph of Section IV-A.

C. Balanced LI Procedure

The unbalanced LI procedure described in Section IV-A can be easily modified to obtain a balanced LI procedure for which no primary/secondary assignment is required or performed. The finite state diagram describing the algorithm performed by each link station is given in Fig. 8. Notice that this procedure essentially consists of two independent executions (one in each direction across the link) of the unbalanced procedure described in Section IV-A. The only difference is the introduction of a wait hold state which guarantees that the two link stations reenter the connected state at about the same time. More specifically, neither station can enter the connected state until both stations have entered the wait hold state. The correctness of this protocol follows directly from the correctness of the unbalanced protocol.

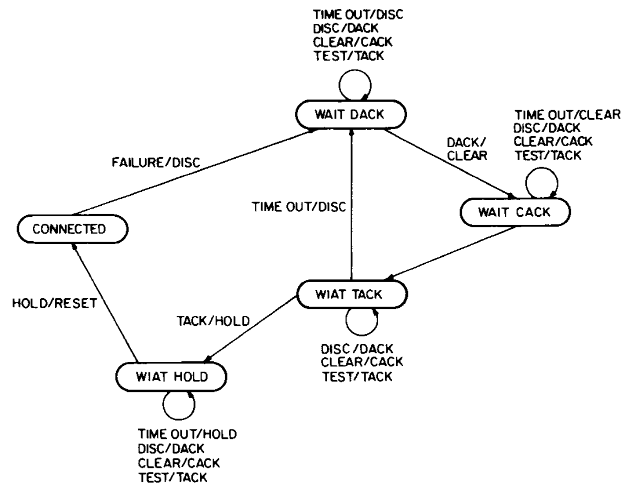


Fig. 8. The new balanced LI procedure.

V. Conclusions

In this paper, we have presented a new class of link initialization procedures that can be used in conjunction with any bit-oriented DLC procedure to achieve data reliability. These LI procedures were shown to ensure synchronization without the use of sequence numbers or time stamps. Moreover, they do not rely on special properties of the physical transmission media such as finite life of messages or line flushing mechanisms. The LI procedures provide, in fact, the flushing mechanism for the entire communication link.

The LI procedures were shown to provide insurance that the link stations do not return to the connected state before the link is capable of reasonable operation. The procedures described in this paper have been designed to comply with the test requirement of one roundtrip transmission within a specified interval. However, they can be easily modified to perform any desired test sequence. The new mechanism can simply be "plugged" into the LI procedure in place of the wait TACK state. On the other hand, if there is no test requirement, the LI procedures may be simplified by removing the TEST and TACK messages and the wait TACK state.

In order to ensure synchronizations across station failures, our LI procedures require one or two bits of nonvolatile memory. One obvious question that arises is whether there exist any LI procedures that can achieve the same goal without nonvolatile storage. Although we have no proof at present, we strongly believe that indeed no such procedure exists.

Before concluding, we note that DLC procedures are often used to provide the lower layer link protocol that serves various distributed network protocols and applications [9], [11], [12]. In this environment, the data sources and sinks of Fig. 1 are the node algorithms that perform the distributed protocols and the packets are the protocol control messages exchanged by the node algorithms. In order to perform correctly, the distributed algorithms require the lower layer link protocols to possess certain properties. These properties are precisely *Delivery, FIFO, Crossing,* and *Deadlock-Free* as defined in this paper. Consequently, window bit-oriented DLC procedure such as HDLC, combined with the LI procedures proposed in this paper, can provide the link-level protocols for distributed network algorithms.

Appendix: Proof of Theorem 2.1

Consider any window-DLC-procedure (i.e., a procedure that has the properties indicated in the first four paragraphs of Section II-B). Suppose that the LI procedure ensures synchronization, i.e., suppose that it has the clear, reset and deadlock-free properties indicated in Section II-C. We need to prove that the DLC procedure has the *Crossing, FIFO,* and *Delivery* properties.

Proof of Crossing

The *clear* property of the LI procedure defines a time t when there are no information frames or information ACK frames in the link and when both DLC processes are in initialization mode. Therefore, at that time in the entire system there are no information frames at all. This means that no frame containing any packet accepted from any source before time t exists in the system. Therefore, no packets accepted from any data source before time t can arrive to any DLC process from the link after that time; hence, we have the *Crossing* property.

Proof of FIFO

Consider the time t defined in the clear property of the LI procedure and suppose that after time t, some DLC process A is the first to go to connected state. The clear property implies that until time t_0 when the other DLC process B also goes to connected state, no information or information ACK frames can be accepted at either process from the other. This is because a) the second process is still in initialization state and it will discard any information frames, and b) the first process will not receive before t_0 information or ack frames because the second does not send any. The *reset* property says that at time t_0, the counter numbers are synchronized, and without loss of generality, we can take them to be 0. For symmetry reasons, it is enough to look at information data flowing in one direction only, from A to B say. Packets accepted by station A from the local data source will be numbered for identification purposes by consecutive increasing number $P(0)$, $P(1)$, $\cdots$ (not modulo W) where $P(0)$ is the first packet accepted after entering connected state. Also, without loss of generality, assume that the sequence number assigned to $P(I)$ is (I mod W).

First observe that the window-DLC mechanism dictates that a frame can be considered acknowledged (and discarded) by A only as a result of receiving an ACK frame and no more than $(W - 1)$ frames can be discarded as a result of receiving a given ACK frame. Also, consecutively discarded frames contain consecutive packets. An ACK frame whose receipt at the sender DLC results in discarded information frames will be called an *active ack*. Observe that an ack is labeled as active or not active only upon being received by A. More precisely, an ACK with number NR received by A is active if $NR \neq VS$, and not active otherwise. The packet corresponding to the *last* discarded frame when an active ack is received will be referred to as *correlated* with the active ack. The *FIFO* property will be proved if we prove the following result.

Lemma A.1: Suppose an ACK arrives at the sender DLC and is labeled active. At the time that the ACK was sent by the receiver DLC, all packets up to and including the correlated packet and only those have been delivered to the data sink in order, with no duplicates and no gaps.

Proof: The following follow directly from the window-DLC properties and will be used in the proof.

a) No frame containing packets prior to and including $P(I - (W - 1))$ can be sent by the sender DLC after accepting packet $P(I)$ and, consequently, no such frame can be received by the receiver DLC after the receipt of any frame containing $P(I)$.

b) No frame containing packets following and including $P(I + (W - 1))$ can be sent by the sender DLC before the time when the frame containing packet $P(I)$ is discarded.

Consider Fig. 9 where $NR1$, $NR2$ denote the ACK numbers of two consecutive active acks. By the window-DLC properties, $NR1 \neq NR2$ and let t_1, t_3 and t_2, t_4 denote the

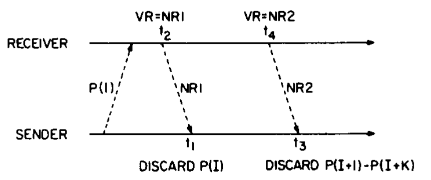


Fig. 9. Timing diagram for proof of Lemma A.1.

respective arrival and departure times. Let $P(I)$, $P(I + K)$ be the packets correlated with the two consecutive active acks. We have $0 < K \leq W - 1$, $I \bmod W = (NR1 - 1) \bmod W$, and $(I + K) \bmod W = (NR2 - 1) \bmod W$. The induction step consists of showing that, if all packets up to and including $P(I)$ and only those have been delivered in order, with no duplicates and no gaps until time t_2, the same is true for all packets up to and including $P(I + K)$ until time t_4. Consider the interval of time from t_2 to t_4. First note that since $P(I + 1)$ is discarded at time t_3, b) above implies that no frame containing packets following and including $P(I + W)$ can arrive at the receiver DLC during the interval $[t_2, t_4]$. Also, a) above says that for all J holds that, after a packet $P(J)$ arrives at the receiver DLC, no frame containing $P(J - (W - 1))$ or preceding packets can arrive at the receiver DLC. Now take J to be $I, I + 1, \cdots, I + (W - 1)$. We obtain that *in the considered interval,* the following must be true: the only frame with send sequence number $NS = NR1$ that can arrive is the one containing $P(I + 1)$; if the frame containing $P(I + 1)$ arrives, the only frame with $NS = (NR1 + 1) \bmod W$ that can arrive afterwards contains $P(I + 2)$; and so on, if the frame containing $P(I + (W - 2))$ arrives, the only frame with $NS = (NR1 + (W - 2)) \bmod W$ that can arrive afterwards contains $P(I + (W - 1))$, and if the latter arrives, no frame with $NS = (NR1 + (W - 1))$ can arrive afterwards. Since the receiver DLC accepts frames only with consecutive send sequence numbers modulo W, only $P(I + 1)$ to $P(I + (W - 1))$ can be accepted in this interval in order and with no gaps or duplicates. Since VR at time t_4 is $NR2$, the packets that have, in fact, been accepted in the interval $[t_2, t_4]$ are $P(I + 1)$ up to and including $P(I + K)$, completing the induction step.

Now, since at time t_0 there are no information ACK frames in the link, the first received active ACK is sent after that time, at time t' say, and let NR' be its ACK number. As before, $NR' \neq 0$ and only frames containing $P(0)$ to $P(W - 2)$ can be received by the receiver DLC. Finally, the same argument as before shows that the ones that have been indeed accepted are $P(0)$ to $P(NR' - 1)$ in order and only once, completing the proof of the lemma and of the FIFO property.

Proof of Delivery

Suppose the sender DLC sends a frame with sequence number VS and let that frame contain packet $P(I)$. From Lemma A.1 follows that when the frame is sent, all packets up to and including $P(I - 1)$ have been already accepted. Therefore, the receiver DLC will not change VR until that packet is correctly received. We have assumed in Section II-B that the probability of frame loss or error is less than 1, and hence, if the frame is repeatedly sent, it will eventually arrive correctly at the receiver DLC. When this happens, the corresponding packet is delivered to the data sink. Similarly, the corresponding ack frame will eventually be received correctly at the sender DLC, causing the acknowledged frames to be discarded and new packets to be accepted from the data source. This completes the proof of the theorem.

References

[1] D. E. Carlson, "Bit-oriented data link control," in *Computer Network Architectures and Protocols,* P. E. Green, Ed. New York: Plenum.

[2] K. A. Bartlett, T. A. Scantlebury, and P. T. Wilkinson, "A note on reliable full-duplex transmission over half-duplex links," *Commun. Ass. Comput. Mach.,* vol. 12, p. 260, May 1969.

[3] ISO/TC9/SC6 N2290, "Data communication—HDLC procedures—consolidation of elements of procedures," ISO/DP 4335.

[4] IBM, "Synchronous data link control," General information, IBM Rep. GA27-3093.

[5] D. Brand and W. H. Joyner, "Verification of HDLC," *IEEE Trans. Commun.,* vol. COM-30, pp. 1136-1143, May 1982.

[6] H. V. Stenning, "A data transfer protocol," *Comput. Networks,* vol. 1, pp. 99-110, 1976.

[7] G. V. Bochmann and R. J. Chung, "A formalized specification of HDLC classes of procedures," in *Proc. NTC'77,* Los Angeles, CA, Dec. 1977, pp. 03A-21.

[8] G. LeLann and H. LeGaff, "Verification and evaluation of communication protocols," *Comput. Networks,* vol. 2, pp. 50-69, 1978.

[9] A. Segall, Distributed network protocols," *IEEE Trans. Inform. Theory,* vol. IT-29, pp. 23-35, Jan. 1983.

[10] Draft IEEE Standard 802.2, "Logical link control, local area network standards," Draft D, Nov. 1982.

[11] A. Segall and J. M. Jaffe, "A reliable distributed route set-up procedure," *IEEE Trans. Commun.,* vol. COM-34, Jan. 1986.

[12] J. Hagouel and M. Schwartz, "Correctness proof of a distributed failsafe route table update algorithm," presented at the 3rd Int. Conf. Distribut. Comput. Syst.", Miami, FL, Oct. 1982.

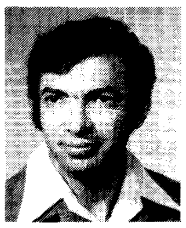
[13] A. V. Shankar and S. S. Lam, "An HDLC protocol specification and its verification using image protocols," *ACM Trans. Comput. Syst.,* vol. 1, pp. 331-368, Nov. 1983.

★

Alan E. Baratz received the B.S. degree in math and computer science from the University of California, Los Angeles, and the M.S. and Ph.D. degrees in computer science from the Massachusetts Institute of Technology, Cambridge, in 1976, 1979, and 1981, respectively. His thesis research with Prof. R. L. Rivest was on algorithms for integrated circuit signal routing.

In 1981 he joined the IBM Thomas J. Watson Research Center, Yorktown Heights, NY, as a Research Staff Member. In 1984 he became Manager of the Telecommunications Network Architecture Project. His research interests include computer communications, distributed algorithms, VLSI layout algorithms, and combinatorial graph algorithms.

★



Adrian Segall (S'71-M'74-SM'79-F'88) was born in Rumania in 1944. He received the B.Sc. and M.Sc. degrees in electrical engineering from the Technion—Israel Institute of Technology, Haifa, in 1965 and 1971, respectively, and the Ph.D. degree in electrical engineering with a Minor in statistics from Stanford University, Stanford, CA, in 1973.

After serving active duty in the Israel Defence Army, he joined in 1968 the Scientific Department of Israel's Ministry of Defence. From 1973 to 1974 he was a Research Engineer at System Control Inc., Palo Alto, CA, and a Lecturer at Stanford University. From 1974 to 1976 he was an Assistant Professor of Electrical Engineering and Computer Science at the Massachusetts Institute of Technology, Cambridge, and in 1976 he joined the faculty of the Technion, where he is now Benjamin Professor of Computer-Communication Networks in the Department of Electrical Engineering. From 1982 to 1984 he was on leave with the IBM T. J. Watson Research Center, Yorktown Heights, NY. His current research interest is in the area of computer communication networks.

Dr. Segall was selected as an IEEE delegate to the 1975 IEEE-USSR Information Theory Workshop, and is the recipient of the 1981 Miriam and Ray Klein Award for Outstanding Research.