AT&T Technical Journal
Vol. 64, No. 2, February 1985
Printed in U.S.A.

Proposed Specification of BX.25 Link Layer Protocol

By R. P. KURSHAN*

(Manuscript received October 3, 1983)

The BX.25 link-layer protocol is a standardized procedure for establishing and maintaining a connection across a data link. Virtually all such standardized communication protocols worldwide are defined through English-language specifications. While the English-language BX.25 specification was developed with exceptional care and in fact represents an improvement over the international standard X.25, it is argued here that it, or any English-language specification, can be expected to fall short of the implicit objective: to define implementations that will be mutually compatible and satisfy given performance criteria. An alternative formal specification is presented. Among its attributes are that it is by its nature mathematically precise; it provides a medium for formal structured development and formal proof of performance of a task (validation) independent of any implementations; and the formal specification may be implemented into software or hardware in an automated fashion, reducing the need for laborious and repetitive implementations and virtually eliminating the need for "certification" of implementations. The specification given here can also be used to resolve issues of vagueness, ambiguity, and contradiction as they arise in the English-language BX.25 specification standard.

I. INTRODUCTION

The BX.25 link-layer (level 2) protocol is for data packet interchange between DTE (Data Terminal Equipment) customer equipment and DCE (Data Circuit-Terminating Equipment) network equip-

* AT&T Bell Laboratories.

Copyright © 1985 AT&T. Photo reproduction for noncommercial use is permitted without payment of royalty provided that each reproduction is done without alteration and that the Journal reference and copyright notice are included on the first page. The title and abstract, but no other portions, of this paper may be copied or distributed royalty free by computer-based and other information-service systems without further permission. Permission to reproduce or republish any other portion of this paper must be obtained from the Editor.

ment, or between two DTEs, over a single data link (a general reference for the terminology used here is Ref. 1). The protocol is designed with the expressed intention to be compatible with the International Standards Organization (ISO) Standard High-level Data-Link Control (HDLC) and International Telegraph and Telephone Consultative Committee (CCITT) Recommendation X.25 Link Access Procedure B (LAPB).

1.1 Function of level 2

The functional purpose of level 2 is to provide end-to-end transmission of "frames" (packets) with the capabilities of error and flow control. Error control is provided through error detection and error correction. Error detection is accomplished by means of a Cyclic Redundancy Check (CRC) for detection of erroneous frames, and by level 2 issued sequence numbers for detection of lost frames. Error correction is accomplished by retransmission of lost or erroneous frames. Flow control is provided through interstation signaling, which interrupts the generation of frames at the remote end. The protocol is defined for use on a synchronous, full-duplex link.

1.2 OSN specification of level 2

This paper is based upon the following document, hereinafter referred to as OSN: "Link-Layer Logical Interface," Section 2, *Operations System Network Protocol Specification: BX.*25, Issue 2, Part II, published by Bell Laboratories in March 1980. (Since the writing of this paper, a third issue has appeared; however, the Part II, Section 2 of concern here remains substantially the same in the new issue.)

1.2.1 Purpose

The purpose of OSN, as described elsewhere in the BX.25 document, is to provide a standard that will serve two goals. The first is to assure compatibility of two stations adhering to the standard. The second is to assure the performance of any implementation adhering to the standard.

1.2.2 Means

The means by which OSN sought to fulfill the above purpose is a very carefully formulated English-language description with numbered paragraphs and associated figures and tables (see OSN) that covers "every aspect" of operation of the protocol. The description is meant to describe the functions of the protocol, independent of any particular implementation.

1.2.3 Critique

The BX.25 link-layer protocol is a standardized procedure for establishing and maintaining a connection across a data link. The protocol is defined through an English-language specification, the way virtually all communication protocols are defined worldwide. In fact, the BX.25 link-layer protocol is a refinement of X.25 LAPB, a protocol adopted by the international telecommunications standards organization, CCITT. BX.25 is compatible with X.25, and corrects certain problems associated with X.25.

Why write a protocol specification? One answer might be to provide implementors with a conceptual sense of the facility to be implemented. This is a reasonable use for such a document, and many of the carefully worded English-language specifications previously adopted as standards, including BX.25, serve fairly well in this capacity.

However, there is no guarantee that an implementation whose design is guided by such a specification will be compatible with any other such implementation. Nor is there a guarantee that any such implementation will maintain any level of performance whatsoever. When stated this way, there are few who would disagree. Nonetheless, there is an implicit inference held by many people that if a communication protocol implementation "adheres" to a specification, then it will be compatible with all other "adhering" implementations and, furthermore, will maintain a certain level of performance. An overriding difficulty with this inference, as we shall see, is that in practice there is virtually never a reasonable criterion for "adherence" of an implementation to an English-language specification.

At the root of the difficulty is the semantic meaning of "specification." Actually, specification would seem to entail a three-fold purpose, which goes far beyond the (certainly useful and viable) role of presenting a conceptual sense of a facility. According to this expanded purpose, a specification first of all should define (logically) a class of implementations (namely, those which adhere to it). Within the context of the specification, it should be possible to state a task that each implementation is intended to perform (e.g., maintain some level of performance). The specification should be such that it is possible, at least in theory, to determine whether it is logically consistent with the stated task (e.g., if part of the required performance is to detect all sequence number errors, it should be possible to determine by theoretical means whether every adhering implementation necessarily will detect all sequence number errors, without actually testing each of what is most likely an infinite number of such possible implementations). If it is not presumed that a task can be stated, or that the question of whether a task always will be performed is logically meaningful, then adherence to a specification loses all practical significance.

The question is whether an English-language specification can satisfy these three points. The claim here is that it cannot. The basic problem with English-language specifications is that they generally lack any concrete form or structure (such as one does find in "formal" specifications). This amorphous quality of English-language specifications leads to a generality which engenders vagueness and ambiguity. Furthermore, a lack of structure fosters the appearance of inconsistencies.

For example, the BX.25 specification is the result of careful thinking by many talented individuals. It is well organized and formulated, probably as much so as any English language specification can be. It represents an improvement over the X.25 specification, which itself was the result of careful thinking by other individuals at the international level; BX.25 itself has undergone two major revisions. All this effort notwithstanding, BX.25 is nonetheless vague (e.g., the conditions under which a poll may be sent are never made explicit), ambiguous (e.g., precedence of the rejection of an out-of-range N(R) described in OSN 2.4.7, over the packet acceptance described in OSN 2.4.5.2, while probably intended, is never stated), and contradictory (e.g., OSN 2.3.4.3 provides for retransmissions of REJ [Reject] while OSN Table 2.1c uses RR [Receiver Ready] consistently in its place). Given the quality effort that went into this specification, it might seem unreasonable to place the blame for these deficiencies upon the individuals who created the specification; an alternative culprit is the specification medium: the nonformal, unstructured English language. This view is reinforced by the (now widely accepted) observation that such difficulties are not the exception but the rule in English-language specifications of communicative protocols.

Definitions that are vague and ambiguous have limited use. It is not surprising to hear reports that different implementations of OSN (and virtually all other standardized communication protocols, for that matter) have been incompatible and of varying performance, even though this was the very situation OSN was intended to prevent.

Let us suppose for the sake of argument that a protocol specification such as OSN were free of imprecision. Where would that leave it with respect to its purpose? How would one determine whether a particular implementation adheres to the specification? Since the specification is nonformal, a formally defined adherence criterion would have no meaning beyond its informal interpretation. Possibly the best adherence criterion would be that several experts examine the implementation and declare that it adheres. The issue of stability of such a criterion aside, however, the consensus of experts is a patently impractical criterion, as many implementations involve optimized code of an arcane nature understood only by its designer. More likely, one would skirt the adherence criterion altogether, relying instead upon testing the implementation. What about testing (which is commonly called "certification"?[2] It is a notoriously difficult problem to design a test for performance of a task. At best, such a test can only say with a certain level of confidence, assuming the implementation has certain properties (which, in fact, we can never know for sure that it does have), that it performs the task. In practice, however, it is virtually never the case that a testing procedure is sufficiently well analyzed that one can assign to the result a qualitatively defined degree of confidence. Thus, the test itself can only suggest that an implementation is okay. Moreover, such a test certainly can say little or nothing about the overall integrity of the specification, since the relationship between the specification and an implementation is not quantized. (As a case in point, there have been satisfactory implementations of BX.25 level 2, the above-mentioned deficiencies in the specification notwithstanding.)

There are, in fact, models that are used for protocol validation, such as Refs. 3 and 4 (more about them later). However, to use such a model, a given specification such as OSN must be translated into the context of the model. As there is no unique way to do this, one might end up with a validated translation of OSN, but certainly not with a validated OSN (other translations may not be valid).

Thus, the status of the English language as a specification medium is this: it does not lend itself to precise, unambiguous, or internally consistent specifications; and there is no meaningful way either to test (abstractly) the properties of a specification or to determine whether a particular implementation adheres to a specification. It does provide a conceptual sense of a facility. However, if this is the only requirement placed on a specification, the specification could be considerably simplified. For example, the concept of the link layer of X.25 is presented in Ref. 1 in a mere four pages (as compared to 27 pages for OSN). When one speaks privately to implementors who have implemented a protocol from an English-language specification, one often learns either they did not concern themselves with the precise details of the specification, or if they did, then these details have led them astray.

1.2.4 This paper

This paper presents an example of an alternative approach to protocol specification. The approach is to present a specification in a formal, mathematically based context that, by its very nature, renders the specification free of vagueness, ambiguity, and inconsistency.

Furthermore, this formal context not only provides a medium for precise specifications, but also provides a medium for a formal, structured development of the protocol and for statement and formal proof of the performance of a task (independent of any implementations). The formal specification may be stored on-line as a database in a computer, and tested and altered quite easily, prior to any implementations. Finally, the specification medium is such that a protocol specification may be implemented into software or hardware in an automated fashion.

This paper presents a formal specification of BX.25 level 2, presented through such a formal specification medium (called the *selection/resolution model for concurrent processes*). The specification is intended to illustrate that such a formal specification can be quite readable (the implementors with whom I have interacted have been able to follow such a specification after about an hour of coaching). A production quality, formal specification of the protocol should be preceded by an informal introduction that presents the concept of the protocol (e.g., the above-mentioned four pages in Ref. 1). It is proposed that such a formal specification with a suitable introduction replace currently accepted English-language specifications. Not only would this permit true validation (formal proof of performance of task); it could also eliminate the need for laborious and repetitive implementations of the same protocol, through application of automated implementation. This would save time and effort, virtually eliminate the need for certification of implementations, and reduce the problem of interimplementation compatibility to the level of hardware interfaces.

The specification given here can also be used to resolve issues of vagueness, ambiguity, and contradiction as they arise in OSN. Deviations from OSN occur only for the purpose of resolving inconsistencies as they appear in that specification; they are indicated by a comment in the right margin of the formal specification. In every case that the formal specification clarifies an ambiguity or resolves a vague issue in OSN, this clarification has been cleared with at least one expert on BX.25 level 2 (such clarifications were too numerous to indicate them in the formal specification).

1.3 Other possible level 2 specifications

There is no dearth of models for specification and analysis of concurrent processes [see the Special Issue on Computer Networks and Protocols of IEEE Trans. Commun., *28*, No. 4 (1980)]. However, there is something unsatisfying about each such model I have examined. Without trying in any way to give a critique of the literature, I will voice my complaints about two such models. The complaints (not the models) may be considered to be representative.

Bochmann and Chung use a Petri net model for a synchronous environment in which transitions are determined by enabling predicates "specified in terms of a high-level programming language such as Pascal."[3] However, this makes no sense in an asynchronous environment as the lower-level interactions of synchronization, interruption, message exchange, and other "critical section" problems are hidden from view by the programming language. This aside, they describe no algorithm for combining several processes, a necessity for any analysis.

Zafiropulo et al. use a finite state model based upon message exchange.[4] While this is (theoretically) capable of specifying processes in asynchronous environments and an algorithm for combining several processes is described, transitions are based simply upon the receipt or transmission of a message. There is no provision for complex interdependencies among processes such as exist in BX.25.

Other specification formats such as Ref. 5 provide actual software to establish a database containing a well-defined, formal specification. However, no tools exist to analyze the resulting protocol.

1.4 The selection/resolution model

The format used here to specify level 2 is based upon a general model for concurrent processes.[6,7] It is arithmetic in nature and has the property that the collective specification and joint behavior of several processes is computed by a fixed algorithm from the specifications of the component processes alone. Processes are described in terms of states and enabling predicates. The states need not be listed exhaustively, but may be represented by one or more parameters. For analysis, a task is formally defined. To facilitate validation (proof that a given task is performed) reduction algorithms are available that substitute for a given system a simpler system with the (provably) same performance. Such reductions are necessary to combat intractability produced by "state explosion."

1.4.1 Methodological attributes

The approach of the selection/resolution model has several methodologically desirable attributes. First, all algorithms may be applied mechanically without understanding the meaning of the protocol specification or task. Second, the procedure for defining the formal specification requires a complete systematic detailing of process behavior, reducing the opportunity, which exists in a less organized approach, to overlook events. The significance of this is not negligible. Most approaches to specification suffer not so much through incorrect assertions as through what is overlooked. While specification mistakes and oversights are still possible in the model described here, they are easily caught (see Section 1.4.2). The given specification is guaranteed to be logically complete in the sense that there can be no unexpected event.

Not only does the specification provide a vehicle for analysis validation, it provides the basis for implementation as a structured program. This is helpful not only to the implementor, whose programming job becomes straightforward if not trivial through automated implementation, it is essential to assure the carry-over of compatibility and performance to each implementation.

1.4.2 Quick review of the model

A system is decomposed into many small interacting automaton-like processes. Suppose process **A** is in an OFF state and process **B** is in its READY state. The process **C** comprised of the two processes **A** and **B** taken together is then in a joint state (OFF, READY). Process **A**, let us suppose, has an ON state and a labelled directed edge from OFF to ON. The label, say, corresponds to the assertion, "**A** is in its OFF state and **B** is in its READY state." This label encodes the condition under which the transition from OFF to ON may occur. It is denoted thus:

$$(\mathbf{A}{:}\text{OFF})\cdot(\mathbf{B}{:}\text{READY})\,.$$

Suppose **B** has a labelled directed edge from READY to its state WILLING with the label

$$(\mathbf{B}{:}\text{READY})\cdot(\mathbf{A}{:}\text{ON})\,.$$

$$\begin{tikzcd} \mathbf{A} & \mathbf{B} \\ \text{OFF} \arrow[d, "(\mathbf{A}{:}\text{OFF})\cdot(\mathbf{B}{:}\text{READY})"] & \text{READY} \arrow[d, "(\mathbf{B}{:}\text{READY})\cdot(\mathbf{A}{:}\text{ON})"] \\ \text{ON} & \text{WILLING} \end{tikzcd}$$

Then the joint process **C** has an "edge" from its state (OFF, READY) to its state (ON, WILLING) with the label

$$(\mathbf{A}{:}\text{OFF})\cdot(\mathbf{B}{:}\text{READY})\cdot(\mathbf{B}{:}\text{READY})\cdot(\mathbf{A}{:}\text{ON})$$
$$(\text{The Boolean product of the label of } \mathbf{A} \text{ and the label of } \mathbf{B})\,.$$

When interpreted as a Boolean expression, this has the logical value 0 (always false) since (**A**:OFF) and (**A**:ON) can never be true simultaneously. Thus, in fact, **C** has no edge between (OFF:READY) and (ON:WILLING). Had **B**'s label been rather

$$(\mathbf{B}{:}\text{READY})\cdot(\mathbf{D}{:}?)$$

then **C**'s edge would have had the (reduced) label

$$(\mathbf{A}{:}\text{ON})\cdot(\mathbf{B}{:}\text{READY})\cdot(\mathbf{D}{:}?)\,.$$

In this example, each process selects (broadcasts to the other processes) the name of its current state. This is a special case of the more general presentation but is typical of the processes defined for level 2.[6]

The joint behavior of all the processes that comprise level 2 taken together is modelled by taking all such possible products as illustrated above. This procedure is curtailed (so as to avoid the need to consider the labels on some 10^{12} possible edges) through formal reductions of the system.[6]

1.5 How the structural organization of the protocol was established

The division of an environment (in this case, level 2) into component processes (see Fig. 1) is part discipline and part art. The general approach involves several passes over a founding document such as OSN, a set of notes or evolving ideas. In the first pass, the so-called explicit processes are identified. These are objects, parameters, routines, and the like that are explicitly named, such as timers, counters, messages, and channels. Some are more obviously translated into processes than others. For example, a counter is a conceptually simple process that advances from state N to state $N + 1$ each time the thing it is counting arrives. A message is produced by a sending process whose states correspond to the message it sends. (The Control Primitive process **CP** sends the primitive REJ when it is in state REJ.) A channel process (such as level 1) is represented as a buffer process which stores the message M it is transmitting while in its state M. Delay is modelled by the length of time the channel process remains in the state M. In this pass, each such explicit process is named and its states are defined (to the extent possible).

Once the explicit processes have been identified, another pass is made over the founding document and the list of explicit processes and associated states. In this pass, implicit processes are identified. These are mainly processes which keep track of what is going on (and hence are dubbed place-keeper processes). For example, if the founding document says, "Transmit SABM and wait for acknowledgement," a place-keeper process must be defined that will (say) start in state 1 and then move to state 2 when SABM (Set Asynchronous Balanced Mode) is sent. This enables the protocol to determine if an appropriate reception is in fact an acknowledgement and not simply out of the blue.

Having identified all the component processes (explicit and implicit) and their respective states, a third pass is made, this time over the processes, to define the selections of each process. A selection is a signal that each process broadcasts from a given state. In level 2 almost every process simply selects the name of the state it is in. In

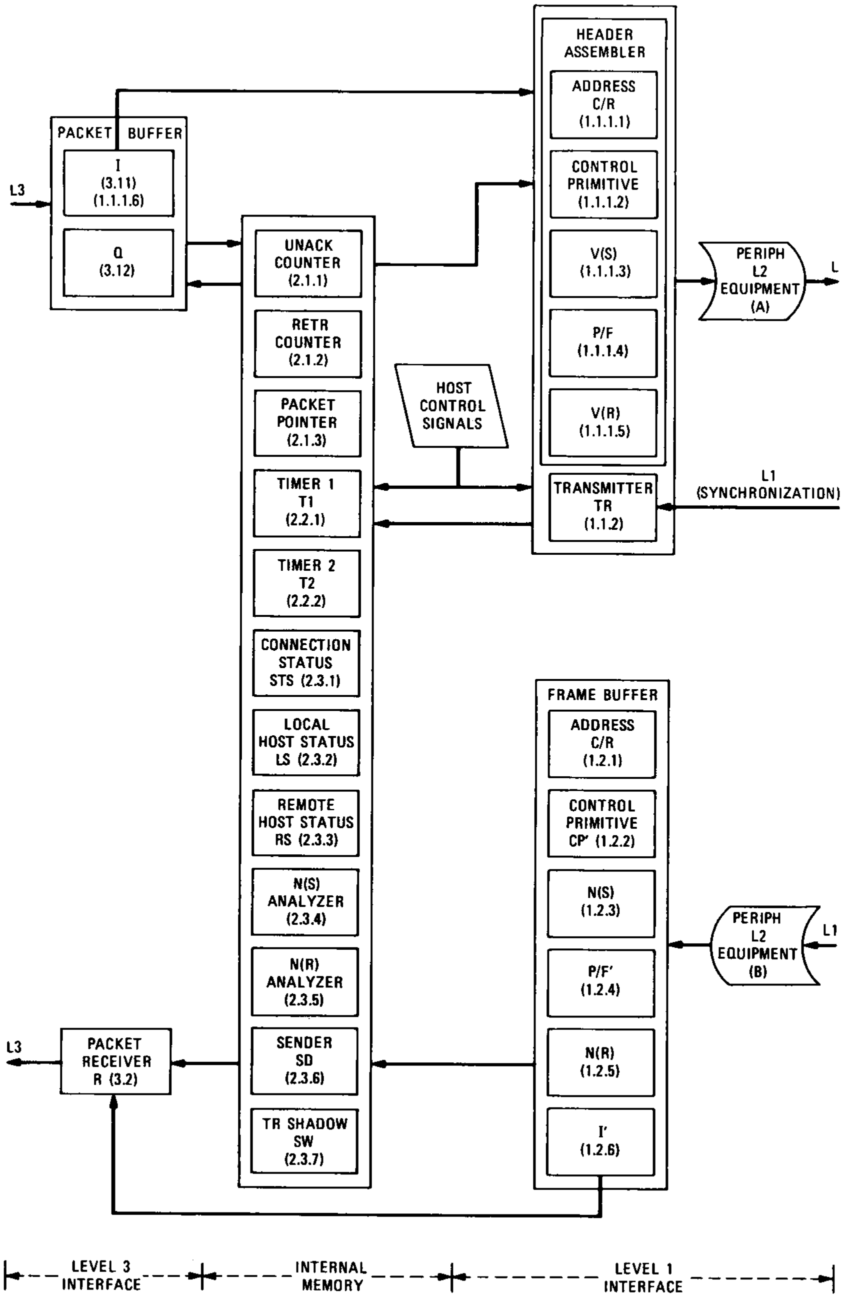

Fig. 1—BX.25 link-layer (level 2) protocol structural organization local end.

other words, it broadcasts the name of its state. (An exception is the transmission process, **TR.**) Finally, a fourth pass is made in which the edge labels of each process are defined in terms of the selections of the various processes. These edge labels are Boolean functions in the selections of various processes that describe the conditions under which the associated transitions occur.

1.6 Sources of errors

Having completed the specification of all the component processes comprising a protocol, one begins a refining procedure to eliminate errors. This is best facilitated when the specification resides in an on-line database. Four types of errors are possible:

1. Errors in the founding document (OSN)
2. Errors of inference from the founding document
3. Errors in transcription (typographical errors)
4. Errors in interpretation of implementation considerations.

The first three types of errors will be caught (insofar as they hinder the formally stated task of the protocol) during the validation phase, during which one applies algorithms to the specification database to prove that the given task is satisfied (or not satisfied). If it is determined that the task is violated, the source of the difficulty is tracked down, corrected, and the validation procedure repeated. This may be continued until a satisfactory specification is found. (Alternatively, there are algorithms for synthesizing a satisfactory specification.[6] However, in the present case, the result would be quite different from OSN.) Aside from verification that the task is performed, it is often useful to examine possible histories of parts of the protocol to determine whether there are any behavior abnormalities (whose exclusion was omitted from the scope of the task).

The fourth type of error is caught upon implementation by the programmer or chip designer who notices that it is impractical or impossible to be faithful to the given specification. This is rectified by altering the specification accordingly and repeating the validation procedure. (An error of the fourth type would not arise if the specification were derived from an actual implementation rather than a conceptual design.)

1.7 Synchronization

The local end (level 2) and remote end are each assumed to be controlled by separate clocks driven by the respective clock of the associated incoming level 1 (see Fig. 2). Thus each level 2 is synchronized to its incoming level 1. The two ends are assumed to synchronize at the interface of level 2 and the outgoing level 1, but are otherwise asynchronous. (This is an inference, nowhere discussed in OSN.)

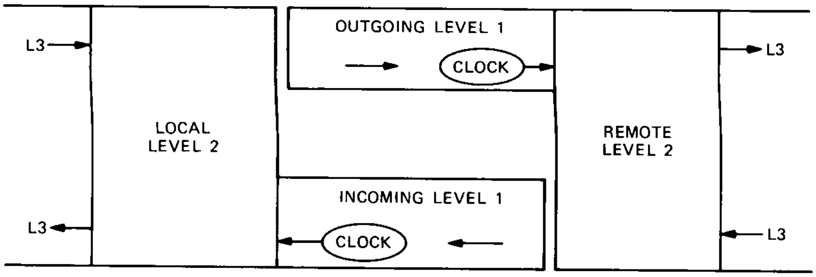

Fig. 2—BX.25 link-layer (level 2) protocol structural organization end-to-end. The two asynchronous components of an end-to-end link layer.

In each synchronous end, time is divided into (not necessarily uniform) intervals. During each time interval (defined separately for each end by the centralized scheduler at each end), each process makes a selection (which in most cases is simply an announcement of its current state). Once each process has done this, and before the end of the time interval, each process resolves the selections of the other processes by moving to a new state along an edge whose label is enabled by the joint selections of the other processes. Once this is done, the time interval is ended. Since different edge labels have varying degrees of complexity, transitions will require varying lengths of time (and the associated time intervals will vary). All this is controlled by the scheduler for the corresponding synchronous end.

However, unlike the specification format,[5] the state transitions all will be relatively quick, and thus it may be desirable to implement the time intervals so all have uniform length. This is especially true if the component processes are implemented in hardware.

1.8 Purpose of this paper

This paper serves to illustrate a specification of BX.25 level 2 using the selection/resolution model.[6,7] The specification was derived almost exclusively from OSN. While every attempt was made to avoid errors in inference, transcription, and interpretation, the greatest likelihood is that all three types will be found here. They, as well as errors in OSN, could be ironed out during validation, which will not be discussed further here.

II. HIERARCHY OF PROCESS GROUPS

The processes that comprise level 2 may be grouped according to function into a hierarchy of classes. At the highest level of the hierarchy, I identify three classes:

1. Level 1 frame interface
2. Internal memory

3. Level 3 packet interface.

The level interface processes transmit or receive, and buffer signals or packets/frames between levels (i.e., packets between level 3 and level 2, and frames between level 2 and level 1). Processes in the internal memory class include counters, timers, and place-keeper processes to remember what is happening, such as "link setup in progress." The three main classes are further subdivided as follows, with process name acronyms indicated in boldface.

1. Level 1 interface
   - 1.1 Outgoing
     - 1.1.1 Header Assembler
       - 1.1.1.1 **C/R** (Address)—identifies frame as command or response.
       - 1.1.1.2 **CP** (Control Primitive)—sets one of I; RR, RNR, REJ, SABM, DISC, DM, UA, FRMR.
       - 1.1.1.3 **V(S)**—sets the "send" sequence number N(S) (generated locally) of the next (information) frame to be transmitted.
       - 1.1.1.4 **P/F**—sets the poll/final bit.
       - 1.1.1.5 **V(R)**—sets N(R), the receive sequence number (generated remotely) of the next expected information frame to be received.
     - 1.1.2 **TR** (Transmitter)—notifies outgoing level 1 and "internal memory" processes of onset of frame transmission.
   - 1.2 Incoming (frame buffer)
     - 1.2.1 **C/R′** (Incoming address buffer)
     - 1.2.2 **CP′** (Incoming control primitive buffer)
     - 1.2.3 **N(S)** (Incoming N(S) sequence number buffer)
     - 1.2.4 **P/F′** (Incoming poll/final bit buffer)
     - 1.2.5 **N(R)** (Incoming N(R) receive sequence number buffer)
     - 1.2.6 **I′** (Incoming packet number buffer)

2. Internal memory
   - 2.1 Counters
     - 2.1.1 **UNACK**—the next expected receive (acknowlegement) sequence number N(R) (sequence generated on this end) in a received (information or supervisory) frame ($0 \leq N(R) < 8$).
     - 2.1.2 **RETR**—number of (re-) transmissions N of a frame subsequent to the expiration of timer **T1** ($0 \leq N \leq N2$).
     - 2.1.3 **PP** (Packet Pointer)—points to packet in packet buffer that was last transmitted to level 1.
   - 2.2 Timers
     - 2.2.1 **T1**—times unacknowledged frame, pending initiation of retransmission procedure.
     - 2.2.2 **T2**—times period of link idleness pending declaration that remote station is unresponsive or inoperative.
   - 2.3 Place-keepers
     - 2.3.1 **STS** (Connection Status—States) S1 = disconnected
       S2 = link setup
       S3 = frame rejected
       S4 = disconnect request
       S5 = information transfer

S6 = REJ primitive sent
S7 = waiting acknowledgement

2.3.2 **LS** (Local host Status)—indicates status of local end.
2.3.3 **RS** (Remote host Status)—indicates busy/not busy status of remote end.
2.3.4 **N(S)A** (N(S) Analyzer)—indicates valid/invalid status of received N(S).
2.3.5 **N(R)A** (N(R) Analyzer)—indicates valid/invalid status of received N(R).
2.3.6 **SD** (Sender)—indicates to-be-sent/already-sent status of header assembler.
2.3.7 **SW**(TR Shadow)—marks the instant that the transmitter TR turns on or off.

3. Level 3 interface

3.1 Packet Buffer—holds packets from local level 3; packet is represented by (local) level 3-assigned packet number.
3.1.1 **I**—indicates number of oldest (local level 3) packet in buffer; in sequence 1, 2, ⋯.
3.1.2 **Q**—indicates queue length in packet buffer.
3.2 **R**(Packet Receiver)—receives packet number of packets received and passed by level 2.

III. PERIPHERAL LEVEL 2 EQUIPMENT

The following pieces of level 2 equipment have no effect on protocol performance (relative to the level 2 task) and hence are not modelled here. Conceptually (and likely physically, too), they sit between the level 2 processes defined above and level 1. Those marked (A) (respectively (B)) sit between the level 2 outgoing (incoming) level 1 interface and outgoing (incoming) level 1.

(A) Flag generator—prefixes every frame with a flag consisting of the sequence 01111110, and generates continuous flags between frames.

(A) Frame checking sequence generator—appends a 16-bit Frame Checking Sequence (FCS) field to the end of every frame that provides a CRC for error detection.

(A) Bit-stuffer—provides for transparency of the flag by inserting a 0-bit after any sequence of five contiguous 1-bits in the address, control and FCS fields.

(A) Frame-abort sequence generator—transmits seven contiguous 1-bits (without 0's) when a frame in the course of transmission is aborted. (See comment in specification of level-1 Transmitter **TR** given in Section VII.)

(B) Idle link detector—detects an input of 15 contiguous 1-bits; when detected, the incoming link is declared idle and timer, **T2**, is set; when a 0-bit is detected, the idle condition is cleared. If **T2** expires in this mode, level 2 **STS** returns to a disconnected state, S1. (This is modelled by a disconnect request DISC retransmitted $N2$ times.)

(B) Invalid frame discarder—discards a frame not bounded by flags,

containing an address field other than 00000001 or 00000011, or having fewer than 32 bits between flags.
(B) Bit-destuffer—reverses bit-stuffing procedure between flags.
(B) Flat sensor—identifies beginning of frame by sensing opening :lag (exactly six contiguous 1's). This facilitates the definition of the address, control, and information fields whose components provide the elements of the frame buffer, and it serves to mark the arrival of a new frame.
(B) CRC decoder—discards a frame which contains errors detected by CRC.
(B) Syntax checker—discards a valid error-free frame which contains any of the following syntactical faults:

- Control field unknown
- Receipt of an information field in a noninformation frame
- Information field exceeds maximum established length
- Receipt of an information field not containing an integral number of octets, if this is not allowed.

When such a frame is detected and discarded, an FRMR primitive is sent in response. (Such faults have no way of being generated w th a valid implementation of levels 2 and 3 [short of the presence of undetected errors introduced by level 1]. However, this process is simulated by the N(R) analyzer process **N(R)A** which causes an FRMR to be sent if the N(R) of an incoming frame is out of range [not between the last received N(R) and the last transmitted sequence number N(S)]. This fault also has no way of being generated by a valid implementation of levels 2 and 3, short of undetected level 1 errors, and serves to represent, for the purpose of end-to-end performance analysis, all the possible syntax faults which result in a FRMR response.)
(B) FRMR information field generator—generates the special information field which is returned with an FRMR frame. (This field has no function with respect to the operation of the protocol. It is used exclusively for diagnostic purposes external to the normal operation of the protocol.)
(B) Frame buffer—buffers the frame address/control fields as they arrive from level 1, via the level 2 peripheral equipment listed above. The syntactically valid address/control fields are separated (by the frame stripper) into the five components listed below, as the bits arrive. Information bits (if any) may also be buffered while level 2 evaluates the (earlier arriving) address/control fields of the frame and determines whether the information field should be passed to level 3 or discarded. (See the packet receiver process **R** described in Section VII.) The address/control fields cannot be evaluated until they have completely arrived and the frame stripper has set the values of the

following five memory elements of the frame buffer:

1. Address
2. Control primitive
3. N(S)
4. P/F
5. N(R).

Once the values of all five elements are set, the frame stripper signals level 2 to evaluate the frame. However, the final determination about the frame cannot be made until the FCS field has been received, the frame has passed the CRC (error check), and the closing flag is checked as valid. (The physical delay associated with filling the frame buffer and checking for frame validity and correctness [CRC] is modelled here as part of the delay in level 1. There is no conceptual difference between a physical implementation in which a frame buffer is sequentially filled and then presented in toto to level 2 for evaluation, and a level 1 that receives the frame, pauses, and then presents the five frame elements [simultaneously] to level 2 for evaluation.)

(B) Frame discarding by peripheral equipment—the effect of frame loss, which results when any of the (unmodelled above-described) peripheral equipment discards a frame, is modelled here by loss of the frame in level 1. Hence, each frame seen by level 2 as modelled here is assumed to be error-free, valid, and syntactically correct, except possibly for an out-of-range N(R).

IV. LEVELS 1 AND 3

In order to analyze the end-to-end peer-level performance of level 2, the channel between the local and remote ends (the physical buyer or level 1) must be modelled. Furthermore, as the end-to-end performance is stated in terms of delivery of level 3 packets, level 3 must be modelled to the extent of packet delivery to the adjoining level 2.

4.1 Level 3

Level 3 is modelled implicitly as presenting two signals to level 2:

- Transmitting packet
- Idle.

These signals are not explicitly represented, but are presented as a nondeterministic filling of the packet buffer. (The actual level 2/level 3 interface is user defined, and may vary from implementation to implementation. However, its definition is not required in order to analyze the end-to-end performance of level 2.)

4.2 Level 1

The end-to-end performance is analyzed in terms of packet throughput in the context of a local level 2, remote level 2, outgoing level 1 (from local end to remote end) and incoming level 1 (from remote end to local end). Of course, it is enough to define one level 2 and one level 1.

Level 1 is modelled as a frame buffer with delay, in order to absorb the delay in level 1/level 2 interfacing devices such as the CRC checker (error detection) and frame format checker. (These are technically part of level 2, but are not specifically modelled here since they do not affect the end-to-end behavior, aside from producing delay). The capacity of level 1 could be modelled as infinity. However, it can be shown on a theoretical basis that it will never hold more than k frames, where k is the maximum number of transmitted I-frames permitted to be unacknowledged by remote level 2. Furthermore, all the delays attributed to level 2 are smaller than the delay of transmitting one frame. (One frame is at least five octets or 40 bits long. For the purpose of this model, the delay of certain level 2 overhead, including error detection and frame format checking, has been attributed to level 1. This contributes a noncumulative delay of no more than the time required to transmit three octets.) Hence, level 1 may be modelled as a buffer with a capacity of only one frame, which is pushed out of the buffer by a nondeterministic selection change after an arbitrary delay. (The model thus models a situation which, in this respect, is more general than reality. There is no harm in doing this if the desired conclusions are nonetheless obtained. Presumably, we do not want a level 2 which is critically sensitive to timing considerations.)

The processes of level 1 are designed as follows:

Outgoing level 1

1. **C/R1** address buffer
2. **CP1** control primitive buffer
3. **N(S)1** N(S) buffer
4. **P/F1** P/F buffer
5. **N(R)1** N(R) buffer
6. **I1** I buffer
7. **R1** Receive moderator (indicates whether level 1 is ready to receive)
8. **S1** Send moderator (indicates whether level 1 is ready to send).

Incoming level 1 component processes are denoted by acronyms followed by a prime (′).

4.3 Synchronization

Incoming level 1 provides clocking to level 2 through its bit stream and flag generation. This establishes bit and octet synchronization with level 2. Hence, the process comprising local levels 2 and 3, the local host and incoming level 1 are all assumed to be synchronized (driven by a common clock), and are modelled as synchronous processes. The *local end* is defined to be (the product of) these processes. Remote levels 2 and 3, the remote host and outgoing level 1 comprise the *remote end*. Thus end-to-end behavior is modelled by these two processes (the local end and the remote end) taken together. The stabilization[7] of each of these two processes represents the two ends as asynchronous (general) processes. Intuitively, the "asynchronous connection" between the local end and the remote end is provided by the transmitter process **TR** of level 2, which synchronizes its transmission to level 1 by sensing the level 1 clock. This is modelled by permitting transmission to level 1 only in the presence of a level 1 selection, "level 1-is-ready" (process **R1**), seen by the transmitter of level 2.

V. LOCAL HOST

The computer comprising the residence of local levels 2 and 3 (local host) may disconnect the node (for preventive maintenance, for example). This is done in an orderly fashion. Before the actual disconnection, local level 2 informs the remote end that a disconnection is in progress. The actual disconnection takes place only once this notice has been acknowledged by the remote end.

Similarly, to reestablish a link, the host causes the level 2 link setup procedure to be initiated.

Furthermore, if local difficulties cause a temporary inability to process incoming frames (for example, a temporary malfunction which causes a buffer overflow condition), the local host may suspend incoming packets by issuing a busy signal. The effect of this is that local level 2 discards any incoming information fields and notifies the remote end of this condition (using the RNR primitive). When the condition terminates, the host clears the busy condition, and level 2 in turn notifies the remote end (with an RR primitive).

The interaction of the local host with local level 2 is modelled by providing the local host (process **H**) with four L-selections visible to local level 2, defined below.

Host L-selection	Interpretation
(**H**:GO)	Local host start command
(**H**:STP)	Local host stop command
(**H**:B)	Station becomes busy
(**H**:NB)	Busy condition clears

VI. SYSTEM PARAMETERS

There are four system parameters upon whose definition this specification depends. (There are other system parameters, such as maximum allowed size of information field or timer periods, upon which this specification does not depend.) The four parameters are:

- Sequence number modulus (set at eight here and in OSN description)
- $N2$—maximum number of (re-) transmissions of a frame subsequent to the exploration of timer **T1**
- k—maximum number of transmitted I-frames permitted to be unacknowledged by remote level 2 ($0 < k < 8$)
- Q_{max}—capacity (in number of packets) of level 3 packet buffer ($0 < Q_{max} < 8$).

VII. PROCESS SPECIFICATION

The *link layer interface* (level 2) is specified as a product of 27 component processes. Additionally, the *physical layer* (level 2) is specified as a product of 16 component processes (8 incoming, 8 outgoing). The processes, outlined in the previous section, are formally specified here.

7.1 Format

The format of specification is as follows. The comments column includes the source in OSN for the given edge labels.

<process acronym> (<process number/name>)

<verbal description>
states: <list of state acronyms followed by (names)>
<comments>

selections: <list of L-selection acronyms followed by (names)>

<state acronym (name) A>→
<state acronym a>: <edge label of the edge (A,a)>
⋮

<state acronym (name) B>→
<state acronym a>: <edge label of the edge (B,a)>
⋮

7.2 Special notation

1. Unlisted edge labels are 0 (no edge).
2. When the list of L-selection acronym (names) is identical to the list of state acronym (names), the former is omitted.
3. When states are identified implicitly in terms of parameters, the range of the parameters is described in the comments column.
4. The word "otherwise" is used to denote the complement of the disjunction (inclusive OR) of the other labels on edges outgoing from

the state in question, multiplied by the sum of the L-selections at that state.

5. $[N]$ denotes the smallest nonnegative integer congruent to N modulo 8 (in particular, $[8] = 0$); $[i \leqslant m \leqslant n]$ denotes $[i] \leqslant [m] < [n]$ if $[i] \leqslant [n]$, and denotes $0 \leqslant [m] \leqslant [n]$ or $[i] \leqslant [m] \leqslant 7$ otherwise.

6. If **A** is a process, its L-selections are expressed as (**A**:S) where S is the selection identifier. The standard relations[6] are satisfied, namely, (**A**:S)(**A**:S′) = 0 if S ≠ S′, (**A**:S) and (**B**:S′) are independent if **A** ≠ **B** and $\sum_S$ (**A**:S) = 1.

7. A prefix dot (●) in edge labels of edges outgoing from a state S of a process **P** denotes the L-selection (**P**:S).

8. We may denote $\sum_{i=l}^{n} (\mathbf{A}{:}S_i) \equiv (\mathbf{A}{:}\{S_l, \cdots, S_n\})$ when convenient.

9. Each process has a single initial state, which is either the state listed first, or in the case of a parameterized state variable, the smallest value of that variable.

7.3 Common elements

The following are definitions of expressions which occur in the edge labels of more than one process. They may be used in implementation to centralize the redundant computations they present. Each expression is defined in terms of a character α(<acronym>), headed in parentheses by the list of the process in which it appears, and meaning, if relevant.

(**C/R**, **PF** Poll may be set)

$$\alpha(\text{poll}) = (\mathbf{N(R)A}{:}\text{OK})[(\mathbf{STS}{:}(5,6,7))[(\mathbf{CP}'{:}\text{I})(\mathbf{LS}{:}\text{NB})(\mathbf{RS}{:}\text{NB}) + (\mathbf{CP}'{:}\{\text{RR}{:}\text{REJ}\})]\cdot(\mathbf{C/R}'{:}\text{C})(\mathbf{P/F}'{:}0) + (\mathbf{C/R}'{:}\text{R})] + (\mathbf{H}{:}\text{GO}) + (\mathbf{LS}{:}\text{STP}) + (\mathbf{RETR}{:}N2) + (\mathbf{T2}{:}\text{EXP})(\mathbf{STS}{:}3)$$

(**CP**, **P/F** A final transmission is pending)

$$\alpha(\text{fp}) = (\mathbf{C/R}{:}\text{R})(\mathbf{P/F}{:}1)(\mathbf{SD}{:}\text{C})$$

(**CP**, **TR**, **SD** All packets in buffer have been sent)

$$\alpha(\text{aps}) = \sum_n (\mathbf{PP}{:}n)(\mathbf{Q}{:}n)$$

(**CP**, **RETR**, **T1**, **I** No new acknowledgment of an outstanding I frame)

$$\alpha(\text{nn}) = \sum_n (\mathbf{UNACK}{:}n)(\mathbf{N(R)}{:}n)$$

(**TR**, **T1** All sent I frames have been acknowledged)

$$\alpha(\text{aa}) = \sum_n (\mathbf{UNACK}{:}n)(\mathbf{V(S)}{:}n)$$

7.4 Level 2 processes

The 27 processes which comprise local level 2 are specified here. The processes which comprise remote level 2 are identical but that to each of their acronyms is adjoined the suffix "r". (This may be seen in the edge labels of incoming level 1.)

For the interconnection of these 27 processes, see Figs. 1 and 3.

C/R (1.1.1.1 address assembler)

Identifies a frame as a command or response by defining the address field. (For command [respectively, response], address is set to that of the remote [local] end.) The edge labels are taken from OSN Table 2.1 and from the need to ensure that a pending "final" (i.e., a response with P/F bit 1) is not superceded as a result of a subsequent frame arrival, as described in OSN sections relating to the poll bit (see description of **P/F** process).

states: C (command)
R (response)

C (command) $\rightarrow$
R:$\cdot\,\tilde{\alpha}$(poll)[($\mathbf{T}_1$:EXP) + (**STS**:3)]
C: otherwise
R (response) $\rightarrow$
C:$\cdot$ [(**P/F**:0) + (**SD**:L)][α(poll) + ($\mathbf{T}_1$:EXP)(**STS**:3)]
R: otherwise

CP (1.1.1.2 Control Primitive Assembler)

Identifies command/response control primitive in the frame control field of the next frame to be sent. This may be updated several times before it is actually sent, if several frames arrive from level 1 during the course of transmission of a long information frame. Level 3 packets and send sequence numbers appear exclusively in frames containing the information control primitive I. Acronyms are those of OSN. Edge labels are from OSN Table 2.1 and 2.4.5.2b.

states:		
I	(Information command)	Frames containing this control primitive are called information frames and contain a send sequence number N(S) and a receive sequence number N(R) in the control field and an information field containing a level 3.
RR	(Receiver Ready)	Indicates local station is ready to receive an I frame; frame contains N(R) in the control field (acknowledging

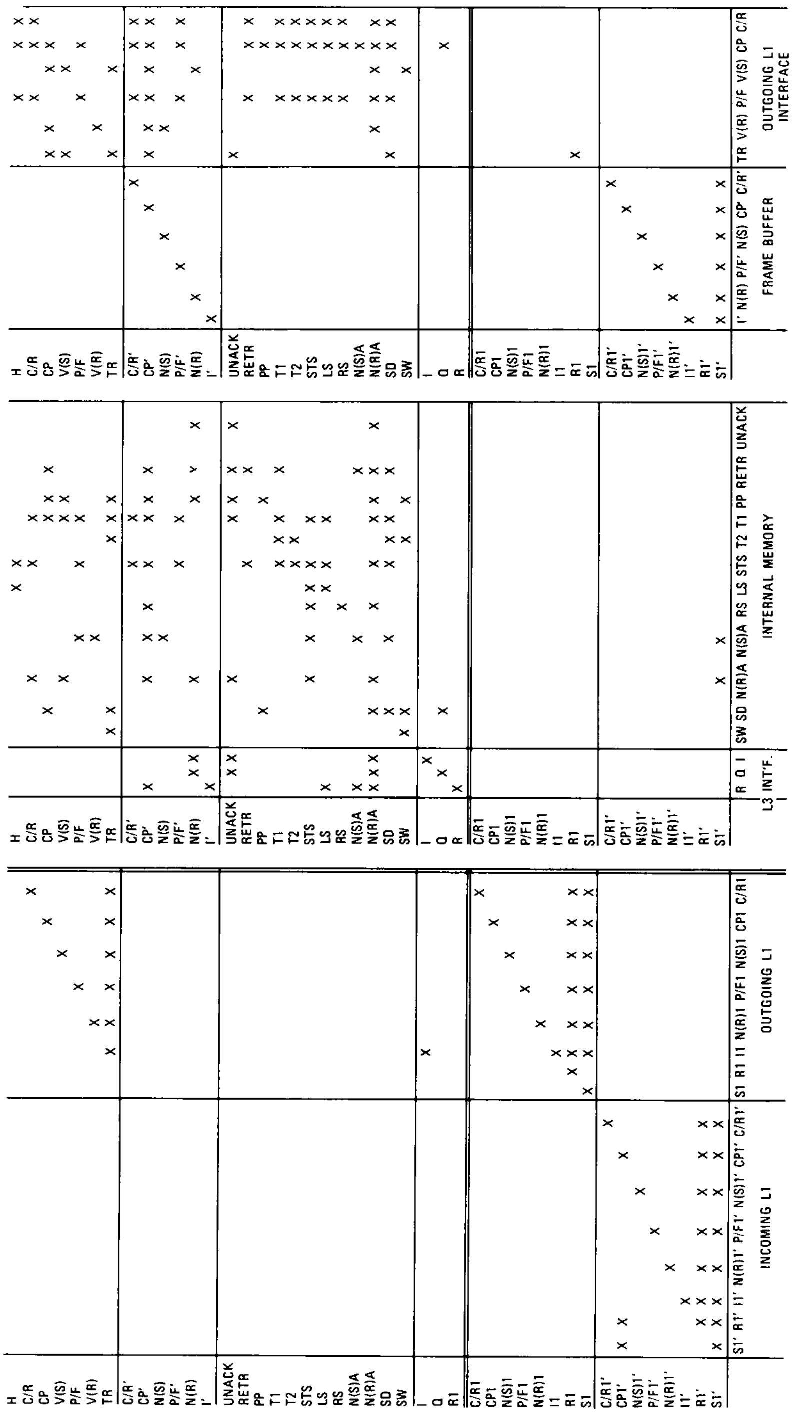

Fig. 3—In the following chart, a process **P** on the horizontal axis "sees" one or more selections of a process **Q** on the vertical axis if the grid point (**P**, **Q**) is marked by an **X**.

		previously sent I frames). “Supervisory” frame.
RNR	(Receiver Not Ready)	Indicates host is busy (unable to accept packets); frame contains N(R). “Supervisory” frame.
REJ	(Reject)	Frame contains N(R) (acknowledging previously sent I frames); constitutes request to retransmit frames with send sequence numbers ⩾ N(R). “Supervisory” frame.
SABM	(Set Asynchronous Balanced Mode Command)	Indicates host wants to set up link.
DISC	(Disconnect Command)	Indicates host is disconnecting from link (e.g., for prevention maintenence).
DM	(Disconnected Mode Response)	Indicates host is disconnected from link.
UA	(Unnumbered Acknowledgment Response)	Indicates receipt of SABM or DISC.
FRMR	(Frame Reject Response)	Indicates receipt of semantically bad frame (e.g., N(R) out of range).

Set α = (**CP**′:I)(**P/F**′:0)(**STS**:{5,7})(**LS**:NB)(**RS**:NB)
  + (**CP**′:{RR,REJ})[(**C/R**′:C)(**P/F**′:0)
  + (**C/R**′:R)](**STS**:{5,6,7}),
β = (**LS**:{NB,B})(**RETR**:N2)(**N(S)A**:OK)(**N(R)A**:OK)
γ = [(**T1**:EXP) + (**C/R**′:R)(**P/F**′:1)](**T2**:EXP). cf: OSN 2.4.5.8

υ(parameter) → $\upsilon \in$ {I.RR, ..., FRMR}

I: ·$\tilde{\alpha}$(fp)$\tilde{\alpha}$(aps)$\alpha\beta\gamma$

RR: ·$\tilde{\alpha}$(fp)β(**STS**:{5,6,7})[γ{(**CP**′:{I,RR,RNR,REJ})(**P/F**′:1)
(**LS**:NB) + (**CP**′:I)(**P/F**′:0)(**LS**:NB)(**RS**:B) + α(aps)α}
+ [(**T1**:EXP) + (**T2**:EXP)][(**STS**:5)(**RS**:NB) +
(**STS**:6)(**RS**:B)](**LS**:NB) + (**T1**:EXP)(**STS**:7)(**LS**:NB)]
The α(aps)α term is from OSN 2.4.5.1.2.

RNR: ·(**RETR**:N2)[(**STS**:{5,6,7})[(**H**:B)(**LS**:NB) +
(**LS**:B)[(**CP**′:1) + (**P/F**′:1)(**CP**′:{RR,RNR,REJ}) +
(**T1**:EXP) + (**T2**:EXP)]]]

REJ: ·$\tilde{\alpha}$(fp)(**LS**:NB)(**RETR**:N2){[(**T1**:EXP) +
(**T2**:EXP)](**STS**:{5,6})(**RS**:NB) +
(**N(S)A**:REJ)(**STS**:5)}

SABM: ·{(**CP**′:DM)(**STS**:1) + (**CP**′:{DM,FRMR})(**STS**:3) +
(**CP**′:{DM,FRMR,UA})(**STS**:{5,6,7}) +

(**T1**:EXP)(**STS**:1) + [(**T1**:EXP) + (**T2**:EXP)](**STS**:2)
+ (**RETR**:N2)(**STS**:{3,5,6,7} + (**H**:GO)}

DISC: ·$\tilde{\alpha}$(fp){(**LS**:STP)(**STS**:{1,4}) + [(**T1**:EXP) + (**T2**:EXP)](**STS**:4)}

DM: ·{(**STS**:1)[(**CP′**:{I,RR,REJ,RNR})(**C/R′**:C)(**P/F′**:1) + (**CP′**:DISC)] + (**STS**:2)(**CP′**:DISC) + (**STS**:4)(**CP′**:SABM)}

UA: ·[(**CP′**:SABM)(**STS**:4) + (**CP′**:DISC)(**STS**:{3,5,6,7})]

FRMR: ·(**N(R)A**:FRMR)

v: otherwise

V(S) (1.1.1.3 sequence number generator)

Indicates the sequence number N(S) (assigned here) in the frame control field of the next information frame to be transmitted. This is "seen" by level 1 (i.e., is part of the frame transmitted to level 1) in I frames only.

states: N (parameter) $0 \leq N < 8$

$N \rightarrow$

$[N + 1]$	·(**SW**:0)(**TR**:ON)(**CP**:I)	OSN 2.3.1.4.1
M:	·(**CP′**:REJ)(**N(R)**:M)(**N(R)A**:OK)	OSN 2.3.5.5, $0 \leq M < 8$
0:	·[(**CO′**:SABM) + (**CP**:SABM)]	if $N \neq 0$; OSN 2.3.3.5
N:	otherwise	

OSN refers to initialization of N(S) only upon receipt of SABM; presumably, this must be done by sending end as well, as modelled here.

P/F (1.1.1.4 poll/final bit assembler)

Indicates value (0 or 1) of poll/final bit in the frame control field of the next frame to be transmitted. (OSN is very vague about this. According to Fred Berg [private communication], the purpose of a poll [P/F bit set to 1 in a command frame] is to force the other end to respond to the associated command, rather than to wait and respond possibly to a subsequent command [or I frame], as is allowed if there is no poll. The response is required to be a "final" [P/F bit set to 1 in a response frame], which indicates it is a response to the outstanding poll, of which there is allowed to be at most one [from each end] at any time. However, when to send a poll is, with one exception, nowhere specified in OSN and thus is left to the implementer. The exception is that when timer T1 expires, a poll is required [OSN 2.4.5.8]. Hence, it must be the intention of OSN that the protocol be equally valid, independent of the extent to which polls are ever used. Furthermore, this polling convention leads to certain inefficiencies. For example, the response during the information transfer to a supervisory poll must be an RR [final], whereas an I frame would convey the same information [less the final bit] more efficiently.) The following sections of OSN mention the P/F bit: 2.3.2, 2.3.3.2, 2.3.3.3, 2.3.3.4, 2.3.3.8, 2.3.4.2, 2.4.3, 2.4.4.4, 2.4.4.6, 2.4.5.5, 2.4.5.7, 2.4.5.8, 2.4.7, 2.4.8.1.

states: 0	Outgoing frame not poll/final.
1	Outgoing frame is poll or final.

υ (parameter) $\rightarrow$ $\upsilon \in \{0,1\}$

1: $\cdot$[(**C/R**′:C)(**P/F**′:1)(**N(R)A**:OK) + α(fp)
+ (**STS**:3)(**T1**:EXP) + α(poll)]

0: $\cdot$[α(fp)[(**C/R**′:R) + (**P/F**′:0)] + α(poll)] α(poll) is added to both edges to model polling conventions for all legal implementations.

TR (1.1.2 transmitter)

Indicates to level 1 the on/off status of the frame transmission. When the transmitter is on, level I receives sequentially the bits comprising a frame, including those of the address/control fields and the information field if the frame is an I frame or FRMR frame. When the transmitter finishes to send a frame, it turns off and one or more flags are then (automatically, by unmodelled peripheral equipment) sent across level 1. The periods during which the transmitter is on can be of variable length, depending upon the length of the information field and the amount of bit-stuffing required. This is modelled by a nondeterministic state change from the "on" states to the OFF state. The lengths of the periods when the transmitter is off must be an integer multiple of the length of time required to send one flag, i.e., one octet (8 bits). This is modelled by a level 1 state change from the "not ready" state to the "ready" state which is assumed to occur at each instant that a complete flat has been sent.

The transmitter process also optionally aborts the transmission of an I frame when a REJ primitive is received concurrently (in an error-free, valid, syntactically correct frame). The delay associated with the transmission of the frame abortion signal by the peripheral frame-abort-sequence generator is modelled here as if absorbed in the delay until the "on" state is next reached.

states:	OFF
	ON/I (transmitting I frame)
	ON (transmitting non-I frame)
	AB (abort)
selections:	(**TR**:OFF)
	(**TR**:ON)
	(**TR**:ON)
	(**TR**:AB)

OFF →
ON/I: ·(**SD**:C)(**CP**:I)(**R1**:1)
ON: ·(**SD**:C)(**CP**:1)(**R1**:R)
OFF: ·otherwise

V(R) (1.1.1.5 received sequence number assembler)

Indicates the sequence number N(S) (assigned at remote end) in frame control field of the next expected information frame to be received. This is "seen" by level 1 (i.e., is a part of the frame transmitted to level 1) in I frames and supervisory frames (RR, RNR, REJ) only.

states: N (parameter)			$0 \leq N < 8$
$N \rightarrow$			
$[N + 1]$	·	(**CO**:I)(**N(S)**:N)(**N(R)A**:OK)	OSN 2.3.1.4.3
0:	·	[(**CP**:SABM) + (**CP**:SABM)]	OSN 2.3.3.5
N:		otherwise	

OSN refers to initialization of V(R) only upon receipt of SABM; presumably, this must be done by sending end as well, as modelled here.

ON/I (transmitting I frame) →		
OFF:	(**TR**:ON)	OSN 2.4.5.5b
AB:	(**TR**:ON)(**N(R)A**:OK) (**CP′**:{REJ,SABM})	Abort is optional.
ON/I:	(**TR**:ON)	OSN does not describe abort upon receipt of SABM: added here.

ON: (transmitting non-I frame) →
OFF: ·
ON: ·
AB: (abort) →
OFF: ·α(aa)
AB: otherwise

C/R′ (1.2.1 incoming address buffer)

Buffers the address field of a frame arriving from level 1.

states: C (command)
R (response)
C (command) →
R:·(**C/R1′**:R)(**S1′**:1)
C: otherwise
R (response) →
C:·(**C/R1′**:C)(**S1′**:1)
R: otherwise

CP′ (1.2.2 incoming control primitive buffer)

Buffers the control primitive in the control field of a frame arriving frame level 1.

states: I	For names of states, see control primitive assembler
RR	process 1.1.1.2.
RNR	
REJ	
SABM	
DISC	
DM	
UA	
FRMR	

v (parameter) $\rightarrow$	$v,w \in \{\text{I},\ldots,\text{FRMR}\}$
w:·(**CP1′**:w)(**S1′**:1)	$w \neq v$
v: otherwise	

N(S) (1.2.3 incoming sequence number buffer)

Buffers the sequence number in the control field of a frame arriving from level 1.

states: N (parameter)	$0 \leqslant N < 8$
N (parameter) $\rightarrow$	$0 \leqslant N,\text{M} < 8$
M (parameter):·(**N(S)1′**:M)(**S1′**:1)	$M \neq N$
N: otherwise	

P/F′ (1.2.4 incoming poll/final bit buffer)

Buffers the poll/final bit in the control field of a frame arriving from level 1.

states: 0
1

v (parameter) $\rightarrow$	$v,w \in \{0,1\}$
w (parameter):·(**P/F1′**:w)(**S1′**:1)	$w \neq v$
v: otherwise	

N(R) (1.2.5 incoming receive sequence number buffer)

Buffers "receive sequence number" in the control field of a frame arriving from level 1.

states: N (parameter)	$0 \leqslant N < 8$
N (parameter) $\rightarrow$	$0 \leqslant N,M < 8$
M (parameter):·(**N(R)1′**:M)(**S1′**:1)	$M \neq N$
N: otherwise	

I′ (1.2.6 incoming information field buffer)

Buffers the packet number of the packet in the information field of a frame arriving from level 1.

states: N (parameter)	$0 < N$
N (parameter) $\rightarrow$	$0 < N,M$
M: (**I1′**:M)(**S1′**:1)	$0 < N,M$
N: otherwise	

UNACK (2.1.1 first unacknowledged frame number counter)

Indicates the next expected receive (i.e., acknowledgment) sequence number N(R) in a received (information or "supervisory") frame. Receipt of the number N(R) = N constitutes acknowledgment from the remote end that the (locally generated) information frame with sequence number N(S) = N was received by the remote end.

states: N (parameter) $0 \leqslant N < 8$

N (parameter) $\rightarrow$ $0 \leqslant N,M < 8$

M (parameter): $\cdot$ **(N(R)**:M)(**N(R)A**:OK) $M \neq N$

N: otherwise

RETR (2.1.2 retransmission counter)

Counts the number of (re-) transmissions of a frame subsequent to the expiration of timer T1. OSN 2.4.5.8 and 2.4.8.2.

states: N (parameter) $0 \leqslant N \leqslant N2$

N (parameter) $\rightarrow$ $0 \leqslant N < N2$

$N + 1$: $\cdot$ (**T1**:EXP)

0: $\cdot$ [(**CP**:{UA,RNR}) +
(**CP′**:{I,RR,RNR,REJ}).
(**N(S)A**:OK)(**N(R)A**:OK)$\tilde{\alpha}$(nn)]

N: otherwise

OSN is vague about what reception is correct enough to warrent resetting RETR to 0 (e.g., RNR is accepted, according to OSN 2.4.5.8, even with (**N(R)A**:FRMR) whereas acceptance of I with (**N(S)A**:REJ) is not indicated positively or negatively).

$N2 \rightarrow$

0: $\cdot$ (**CP**:SABM)(**SD**:L)

$N2$: otherwise

PP (2.1.3 packet pointer)

Points to packet in packet buffer that was last transmitted to level 1. (This process "goes back N" for retransmission of lost packets. When acknowledgments arrive, it must decrement accordingly, to keep pace with the packet buffer queue length, process **Q**.)

states: N (parameter) $0 \leqslant N \leqslant Q_{\max}$

$N \rightarrow$ $0 \leqslant N \leqslant Q_{\max}$

$N + 1$: $\cdot$(**SW**:0)(**TR**:ON)(**CP**:I) if $N < Q_{\max}$

M: $\cdot$(**N(R)A**:OK) $\sum_{[j-i]=N-M}$ {(**N(R)**:i)(**UNACK**:j)
+ (**CP′**:REJ)(**N(R)**:i)(**V(S)**:j)}

N: otherwise

T1 (2.2.1 timer 1)

Times unacknowledged frame, pending initiation of retransmission

procedure. (Expiration of timer is modelled as a nondeterministic selection change, and thus is more general than reality.)
states: OFF
ON
EXP (expired)

Define
α = (**CP**:{REJ,SABM,DISC,FRMR}) + (**C/R**:C)(**CP**:RR) + $\tilde{\alpha}$(aa) + (**C/R**:C)(**P/F**:1)

OFF →
ON: · α
OFF: · [$\tilde{\alpha}$ + (**CO**:FRMR)] OSN 2.4.4.1, 2.4.4.3, 2.3.3.3, 2.3.4.3, 2.4.5.4, 2.4.8.1 and 2.4.6, in which setting T1 when sending FRMR is unformal. In part, α is defined in accordance with the comment to state 6 of STS.

Define
β = (**STS**:{2,4})(**CP′**:UA) + (**N(R)A**:OK)(**CP′**:{RR,RNR,REJ}) (**C/R′**:R)(**P/F′**:1) + $\tilde{\alpha}$(nn)

ON →
OFF: · β OSN 2.4.4.1, 2.4.5.4, 2.4.5.8
EXP: · $\tilde{\beta}$
ON: · $\tilde{\beta}$

EXP →
OFF: · (**TR**:ON)(**SD**:L)[(**STS**:{1,2})(**CP**:SABM) + (**STS**:3)(**CP**:FRMR) + (**STS**:4)(**CP**:DISC) + (**STS**:{5,7})(**LS**:NB)(**CP**:RR) + (**STS**:6)(**LS** < NB)(**CP**:REJ) + (**STS**:{5,6,7})(**LS**:B)(**CP**:RNR) + (**N(R)A**:OK)(**P/F′**:1)(**C/R′**:R)$\tilde{\alpha}$(nn)]
EXP: otherwise

T2 (2.2.2 timer 2)

Times period of "link idleness" and periods during which consecutive flags are received. (Expiration of timer is modelled as a nondeterministic selection change, and this is more general than reality.)
states: OFF
ON
EXP (expired)

OFF →
ON: ·(**T1**:OFF)
OFF: otherwise

ON → It is assumed that T2 times any combination of link idleness and consecutive flags (OSN is vague about this).

If the local end rejects a frame while the remote end enters a failure mode wherein it transmits continuous flags (only), the local end will eventually wait forever in **STS** state 7, without any further transmissions.

EXP: ·(**T1**:ON)(**TR**:OFF)
OFF: ·(**T1**:ON) + (**TR**:OFF)]
ON: ·(**T1**:ON)(**TR**:OFF)
EXP →
OFF: ·(**TR**:OFF)(**SW**:0)(**SD**:L)
EXP: otherwise

STS (2.3.1 connection status)

Indicates status of level 2 link control. Acronyms, names, and definition are from OSN Table 2.1. (Collisions of conflicting commands from remote end, host or internal devices such as timers are rarely resolved in OSN; here, in each case a [hopefully] reasonable resolution is given.)

states: 1 (disconnected)
2 (link setup)
3 (frame rejected)
4 (disconnect request)
5 (information transfer)
6 (REJ control primitive sent)
7 (waiting acknowledgment)

1 (disconnected) →
2:·[(**CP′**:DM) + (**T1**:EXP) + (**H**:GO)](**CP′**:SABM)
5:·(**CP′**:SABM)
1: otherwise
2 (link setup) →
1:·(**CP′**:{DM,DISC})(**LS**:STP)
4:·(**LS**:STP)(**CP′**:UA)
5:·(**CP′**:UA)
2: otherwise
3 (frame rejected) →
1:·(**CP′**:DISC)
2:·[(**CP′**:{DM,FRMR}) + (**RETR**:*N*2) +
(**H**:GO)](**LS**:STP)(**CP′**:{SABM,DISC})
4:·(**LS**:STP)(**CP′**:{SABM,DISC})
5: (**CP′**:SABM)
3: otherwise
4 (disconnect request) →
1:·[(**CP′**:{UA:DM}) + (**RETR**:N2)](**H**:GO)
2:·(**H**:GO)
4: otherwise

Set α = [(**CP**′:{UA,DM,FRMR}) + (**RETR**:N2) + (**H**:GO)]
·(**CP**′:DISC)(**N(R)A**:FRMR)[(**LS**:STP) + α(fp)].
β = (**N(R)A**:FRMR)(**CP**′:DISC),
γ = $\tilde{\alpha}$(fp)(**LS**:STP)(**CP**′:DISC)(**N(R)A**:FRMR),
δ = $\tilde{\alpha}$(fp)[(**T1**:EXP) + (**T2**:EXP)](**CP**′:{DISC,UA,DM,FRMR})
(**RETR**:NZ)(**H**:GO)(**N(R)A**:FRMR)(**LS**:STP),
ϵ = $\tilde{\alpha}$(fp)(**LS**:NB)(**N(S)A**:REJ)(**N(R)A**:OK)

5 (information transfer) →
1:·(**CP**′:DISC)
2:·$\alpha\tilde{\epsilon}$
3:·β
4:·γ
6:·ϵ(**CP**′:DISC)
7:·(**LS**:NB)$\delta\tilde{\epsilon}$
5: otherwise

6: (REJ control primitive sent) →
1:·(**CP**′:DISC)
2:·α(**CP**′:{I,SABM})
3:·β
4:·γ
5:·(**LS**:NB)[$\tilde{\alpha}$(fp)(**CP**′:I) + (**CP**′:SABM)].
(**N(R)A**:FRMR)(**CP**′:DISC)
7:·δ(**CP**′:{I,SABM})
6: otherwise

Modelled according to OSN Table 2.1.c which permits at most 1 retransmission of REJ, this contradicts OSN 2.3.4.3 which permits up to N2 retransmission of REJ. Here, RR rather than REJ is retransmitted upon expiration of T1, up to N2 times.

7 (waiting acknowledgment) →
1:·(**CP**′:DISC)
2:·α(**CP**′:SABM)[(**CP**′:{RR,RNR,REJ}) + (**C/R**′:C) + (**P/F**′:0)]
3:·β
4:·γ
5:·[(**CP**′:SABM) + [$\tilde{\alpha}$(fp)(**CP**′:{RR,REJ}) + (**CP**′:RNR)]
(**C/R**′:R)(**P/F**′:1)]
7: otherwise

LS (2.3.2 local host status)

Indicates status of local end as deduced from last received control signal from the local host. OSN Table 2.1.

states:	NB	(not busy)
	B	(busy)
	STP	(stop)

OFF
NB (not busy) →
B: ·(**H**:B)
STP:·(**H**:STP)
NB: otherwise
B (busy) →
NB: ·(**H**:NB)
STP:·(**H**:STP)
B: otherwise
STP (stop) →
OFF:·(**STS**:1)
NB ·(**H**:GO)
STP: otherwise
OFF →
NB: ·(**H**:GO)
OFF: otherwise

RS (2.3.2 remote host status)

Indicates busy/not busy status of remote end as deduced from a receipt of RNR control primitive from remote level 2, based upon OSN Table 2.1.

states: NB (not busy)
B (busy)
NB (not busy) →
B: ·(**CP**′:RNR)(**N**(**R**)**A**:OK)(**STS**:{5,6,7})
NB: otherwise
B (busy) →
NB:·(**CP**′:{RR,REJ,SABM})(**N**(**R**)**A**:OK)
B: otherwise

N(S)A (2.3.4 N(S) analyzer)

Indicates whether or not the sequence number N(S) of the last received (information) frame is the next expected sequence number. If not, it causes a retransmission request (REJ primitive) to be sent.

OSN 2.3.4.2

states: OK
TEST
REJ
OK →
TEST:·(**S1**′:1)
OK: otherwise
TEST →
OK:·[(**STS**:{5,6,7}) + (**CP**′:1) + $\sum_n$ (**V**(**R**):n)(**N**(**S**):n)]
REJ:otherwise
REJ →

OK: $\cdot\tilde{\alpha}$(fp)
REJ: $\cdot\alpha$(fp)

N(R)A (2.3.5 N(R) analyzer)

Indicates whether or not receive sequence number N(R) in the last received (information or "supervisory") frame is "within range," i.e., whether or not [UNACK* ≤ N(R)* ≤ V(S)*], where (*) denotes the state of the associated counter. An N(R) is within range if and only if it acknowledges a sent but yet unacknowledged frame (in which case it also acknowledges all previous unacknowledged frames). If N(R) is not within range, the N(R) analyzer process **N(R)A** causes the link to be reinitialized (the FRMR primitive is sent), and all previously transmitted but unacknowledged I frames are cleared from the packet buffer (they remain unacknowledged and are not retransmitted). (According to OSN 2.3.3.5, detection and recovery from the possible loss of such frames is left to a higher-level protocol. I find no reason not to provide for recovery in level 2. This could be accomplished merely by retransmitting all frames starting with the first unacknowledged one, without reinitializing the link. However, the specification given below is consistent with OSN 2.3.3.5.)

states: OK (within range)
TEST
FRMR (not within range)

OK (within range) →
TEST: $\cdot$(**S1**′:1)
OK: otherwise
TEST →
OK: $\cdot$[(**STS**:{5,6,7}) + (**CP**′:{I;RR,RNR,REJ}) $+ \sum_{[i \le n \le j]}$ (**UNACK**:i)(**N(R)**:n)(**V(S)**:j)] OSN 2.4.7c
FRMR: otherwise
FRMR →
OK: $\cdot$(**STS**:3)
FRMR: otherwise

SD (2.3.6 sender)

Classifies current status of header assembler as "to be sent" or "already sent." (This is not an explicit process of OSN; it is used in conjunction with the transmitter process **TR** to coordinate buffering of outgoing frame.)

states:	L (last)	Header already sent.
	L′	Placekeeping state.
	C (continuing)	Header to be sent.

selections: (**SD**:L)
(**SD**:C)

L (last) $\rightarrow$
L$'$:$\cdot$[(**N**(**R**)**A**:TEST) + $\tilde{\alpha}$(aps)(**CP**:1)]
L: otherwise
L$'$ $\rightarrow$
C: (**SD**:L)(**N**(**R**)**A**:TEST)[α(aps) + (**CP**:1)]
L$'$: (**SD**:L)[(**N**(**R**)**A**:TEST) + $\tilde{\alpha}$(aps)(**CP**:I)]
C (continuing) $\rightarrow$
L:$\cdot$(**TR**:ON)(**SW**:0)
C: otherwise

SW (2.3.7 TR shadow)

Marks the instant that the transmitter process **TR** turns on and off. (Implicit process for control of certain counter.)
states: 0 (transmitter was OFF)
    1 (transmitter was not OFF)
0 (transmitter was OFF) $\rightarrow$
1: $\cdot$ (**TR**:OFF)
0: otherwise
1 (transmitter was not OFF) $\rightarrow$
0: $\cdot$ (**TR**:OFF)
1: $\cdot$ otherwise

I (3.1.1 packet counter)

Indicates the smallest in-sequence level 3-assigned number of a packet unacknowledged by the remote end.
states: N (parameter) $N \geqslant 1$
$N \rightarrow$
$N+1$:$\cdot$(**N**(**R**)**A**:OK)$\tilde{\alpha}$(nn)
N: otherwise

Q (3.1.2 packet buffer queue length)

Indicates the number of packets N in the packet buffer. (By assumption, the packets in the packet buffer bear numbers M, $M+1$, $\ldots$, $M+N-1$ where M is the state of the packet counter **I**.) The buffer may be loaded by local level 3; this is modelled by a nondeterministic jump in state from N to $N+1$.
states: N (parameter) $0 \leqslant N \leqslant Q_{\max}$
$N \rightarrow$
$N+1$:$\cdot$ $N < Q_{\max}$
M: $\cdot$ (**N**(**R**)**A**:OK) $\sum_{[j-i]=N-M}$ (**N**(**R**):i)(**UNACK**:j) $0 \leqslant M < N$
N: $\cdot$ (**N**(**R**)**A**:OK)
$N \rightarrow$
$N+1$:$\cdot$ if $0 \leqslant N < Q_{\max}$

$M+1$:· (N(R)A:OK) $\sum_{[j-i]=N-M}$ (N(R):i)(UNACK:j) — if $0 \leq M < N-1$

M: · (N(R)A:OK) $\sum_{[j-i]=N-M}$ (N(R):i)(UNACK:j) — if $0 \leq M < N$

N: otherwise — $0 \leq N \leq Q_{\max}$

The transition from N to $M+1$ models the simultaneous acknowledgment of $N - M$ frames and the reception of a frame from level 3.

R (3.2.1 packet receiver)

Indicates the level 3 packet number of last packet passed by level 2 to level 3. (The passing of the actual packets is modelled here by the passing of these packet numbers. The transmission of bits from level 1 to level 3 is assumed to occur in real time. That is, it is assumed that level 2 can transmit to level 3 as fast as level 1 can transmit to level 2. The sequence of events during the course of which a packet passes from level 1 [through level 2] to level 3 is as follows. An incoming information frame is first analyzed by peripheral level 2 equipment (B). If the frame is not immediately discarded on the basis of an invalid opening flag on syntactically bad address/control fields, the control field is analyzed by the N(S)/N(R) analyzers. The control field is followed immediately by the information field [packet]. Two possible ways to treat this packet are (1) to immediately transmit it to level 3 and then abort this transmission if the level 2 analysis indicates the frame must be discarded or (2) to buffer the packet during the course of the level 2 analysis. [In option (2), virtually the entire frame must be buffered, as the CRC check and closing-flag-valid check cannot be made until the entire frame has been received.] In either case, the checks on the frame are not all simultaneous. When the final check test is passed [valid closing flag] the UNACK and V(R) counters are updated.)

states: N (parameter) — $N \geq O$

$N \rightarrow$

M ·(CP′:I)(N(S)A:OK)(N(R)A:OK)(LS:NB)(I′:M) — $M \neq N$

N: otherwise

7.5 Level 1 processes

The eight processes comprising incoming level 1 are modelled here. Each process has an acronym which ends with a prime (′). The processes of outgoing level 1 are identical but that their acronyms omit the prime (′).

C/R′ (4.1 incoming L1 address)

Signals the address field of a frame arriving from remote level 2.

states: C
R
ϕ (null state)

v (parameter) → $\quad v \in \{C,R\}$
ϕ:·(**S1**′:1)
v: otherwise

ϕ (null state) $\quad v \in \{C,R\}$
v ·(**R1**′:1)(**TR**r:ON)(**C/R**r:v)
ϕ: otherwise

CP1′ (incoming control primitive)

Signals the control primitive in the control field of a frame arriving from remote level 2.

states: I — For names of states, see control primitive assembler process 1.1.1.2.
RR
RNR
REJ
SABM
DISC
DM
UA
FRMR
ϕ (null state)

v (parameter) → $\quad v \in \{I,...,FRMR\}$
ϕ:·(**S1**′:1)
v: otherwise

ϕ (null state) →
v:·(**R1**′:1)(**TR**r:ON)(**CP**r:v) $\quad v \in \{I,...,FRMR\}$
ϕ: otherwise

N(S)1′ (4.3 incoming L1 sequence number)

Signals the sequence number in the control field of a frame arriving from remote level 2. (The state of **N(S)** is equal to the state of **V(S)** when the frame was transmitted).

states: N (parameter) $\quad 0 \leq N < 8$
ϕ (null state)

N (parameter) →
ϕ ·(**S1**′:1)
N: otherwise

ϕ (null state) →
N·(**R1**′:1)(**TR**r:ON)(**V(S)**r:N)(**CP**r:I)
ϕ: otherwise

P/F1′ (4.4 incoming L1 poll/final bit)

Signals the poll/final bit in the control field of a frame arriving from remote level 2.

states: 0
        1
        ϕ (null state)

v (parameter) → $\qquad\qquad$ $v \in \{0,1\}$
$\phi \cdot$(**S1′**:1)
v: otherwise
ϕ (null state) →
v:·(**R1′**:1)(**TR**r:ON)(**P/F**r:v)
ϕ: otherwise

N(R)1′ (4.5 incoming L1 receiver sequence number)

Signals the state of the received sequence number assembler **V(R)** in the control field of a frame arriving from remote level 2.

states: N (parameter) $\qquad\qquad$ $0 \leqslant N < 8$
        ϕ (null state)

N (parameter) →
ϕ:·(**S1′**:1)
N: other
ϕ (null state)
N:·(**R1′**:1)(**TR**r:ON)(**V(R)**r:N)(**CP**r:{I,RR,RNR,REJ})
ϕ: otherwise

I1′ (4.6 incoming L1 information field)

Signals the packet number of the packet in the information field of a frame arriving from remote level 2.

states: N (parameter) $\qquad\qquad$ $0 < N$
        ϕ (null state)

N (parameter) →
ϕ:·(**S1′**:1)
N: other
ϕ (null state) →
N:·(**R1′**:1)(**TR**r:ON)(**I**r:N)(**CP**r:1)
ϕ: other

R1′ (4.7 incoming L1 receive moderator)

Indicates to remote level 2 whether or not incoming level 1 is ready to receive a frame. (This process provides synchronization for transmissions from remote end to local end.)

states: 0 (not ready to receive)
        1 (ready to receive)

0 (not ready to receive) →
1:·(**CP1′**:ϕ)
0:·
1 (ready to receive) →

0:·(**TR**r:OFF)(**SW**r:1)
1: otherwise

S1′ (4.8 incoming L1 send moderator)

Indicates to local level 2 whether or not level 1 is ready to send a frame.

states: 0 (not ready to send)
        1 (ready to receive)

0 (not ready to receive) →
1:·(**CP1**:ϕ)
0:·
1 (ready to receive) →
0:·

VIII. ACKNOWLEDGMENTS

I wish to thank Fred Berg and Paul Long for their generous contributions to my understanding of BX.25, which understanding, if still imperfect, reflects only upon myself.

REFERENCES

1. A. S. Tanenbaum, *Computer Networks*, Englewood Cliffs, N.J.: Prentice-Hall, 1981.
2. K. Bartlett and D. Rayner, "The Certification of Data Communication Protocols," Proc. Trends and Applications Symp., NBS, 1980.
3. G. V. Bochmann and R. J. Chung, "A Formalized Specification of HDLC Classes of Procedures," Proc. NTC 1977, Los Angeles, 3A2-1/11.
4. P. Zafiropulo, et al., "Towards Analyzing and Synthesizing Protocols," IEEE Trans. Commun., *COM-28* (1980), pp. 651–60.
5. T. P. Blumer and R. L. Tenney, "A Formal Specification Technique and Implementation Method for Protocols," Tech. Report No. 81-15, Bolt, Beranek, and Newman, Inc., 1981.
6. R. P. Kurshan, "Coordination of Concurrent Probabalistic Processes," Lect. Notes in Contr. Inf. Sci., *58* (1984), pp. 605–14.
7. S. Aggarwal, R. P. Kurshan, and K. Sabnani, "A Calculus for Protocol Specification and Validation," Proc. IFIP Protocol Spec., Test., Verif., North-Holland (1983), pp. 19–34.

AUTHOR

Robert P. Kurshan, Ph.D. (Mathematics), 1968, University of Washington; Krantzberg Chair for Visiting Scientists, Technion, Haifa, Israel, March–August 1976; AT&T Bell Laboratories, 1968—. Mr. Kurshan is a member of the Mathematics Research Center. His current interests concern formal development, specification, and analysis of distributed systems such as communication protocols.