Study on Data Network Protocols: Phase I

Formalized Methods for Protocol and Interface Descriptions

Final Report

DÉPARTEMENT D'INFORMATIQUE

Faculté des arts et des sciences
Université de Montréal
C.P. 6128 Montréal 101

checked 11/83

Study on Data Network Protocols: Phase I

Formalized Methods for Protocol and Interface Descriptions

Final Report

by

G.V. Bochmann

Département d'informatique et de recherche opérationnelle

Université de Montréal

February 1979

This final report on data network protocols (phase I) is respectfully submitted to the Department of Communications as requested and in accordance with contract 04SU.36100-8-9523 between the Department of Communications and the University of Montreal. The work was carried out by G.V. Bochmann under the scientific supervision of Y.F. Lum. Appreciation is extended to many individuals whose comments provided valuable input to this work.

Opinions expressed in this report are those of the author.
They do not imply any position of the Department of Communications.

TABLE OF CONTENT

1. Introduction

Computer communication protocols are used for data communications and distributed data processing. The widespread application of such systems and the increasing need for interconnection and interchangeability of system components makes compatibility an important issue. During the elaboration of protocol standards for data communication, the CCITT and the subcommittees SC 6 and SC 16 of ISO/TC 97 have identified the need for more precise protocol description methods. A precise description of a protocol is needed, during its design, as a reference document for the analysis of its correctness and efficiency. The same reference document may serve later as the basis for the implementation of the protocol in different systems. The description of the protocol standard is used, in addition, to judge whether a given protocol implementation meets the requirements of the standard. Since a protocol description in plain language, as used for most existing standards, is usually not very precise, often incomplete, or contains ambiguities, it is not the best candidate for the reference document mentioned. Formal protocol description methods have been developed to overcome the difficulties of natural language descriptions [1].

Already some kinds of state diagrams have been used for describing formally some aspects of certain protocol standards [2, 3, 4]. These approaches to more precise protocol descriptions seem to go in the right direction, although in the case of the packet level procedures of X.25 some difficulties have been pointed out for the state diagram approach taken [6, 7].

The objective of this study is to advance the application of formal description methods for obtaining precise specification of protocol standards, and in particular, to determine a formal description method based on state diagrams suitable for use in ISO and CCITT, taking into account existing description methods.

2. Results of the Study

The main result of the study are the contributions to ISO and CCITT given in the Appendices and commented on below. They contain the following three points which go beyond previous work in the area:

(a) Principles for the specification of communication services and protocols: Within the context of a layered architecture of communication systems, as developed in the ISO Subcommittee on "Open Systems Interworking" (TC 97/SC 16) and the CCITT Special Rapporteur's Group on "Layered Models of Public Data Network Services Applictions" (Study Group VII), the importance of service specifications and formal protocol specifications, and their relationship within the architecture, have been pointed out. (See Appendices 1, 2 and 4).

(b) Towards a language for formal protocol specifications:
The contributions point out a general method, and an approach to developing a language for describing protocols, based on the different approaches to formal descriptions mentioned above. This method is demonstrated by a formal description of the link set-up and clearing procedures of the LAP B of X.25 (level 2). (See Appendices 1, 3 and 4).

(c) Communication service descriptions: A possible method for formally specifying communication services is presented. It is demonstrated by a description of the link layer service provided by an HDLC protocol (see Appendices 1, 2 and 4). We note that the method has also been used for describing the service of a transport protocol [8, 9].

The paper of Appendix 1 was submitted to ISO/TC 97/SC 6 Working Group 1 meeting in February 1979, as a follow-up to previous contributions from Canada [4] and Germany [5] on the state diagram description of HDLC.

The papers of the Appendices 2 and 3 were submitted to the CCITT/SGVII Special Rapporteur's meeting on "Layered Models..." in February 1979.

The paper of Appendix 4 is a Canadian contribution to ISO/TC 97/SG 16. It is written in the form of a comment on the Reference Model for Open Systems Architecture (SC 16 N 117, November 1978) and suggests improvements to the text of its Annex E on "Fromal Description Techniques".

3. Conclusions and future developments

As a general conclusion we note that the results of this study represent some small and, hopefully, useful contribution on a long way to go. Although most individuals and groups involved in the design of communication protocols and standards find the work on formal description techniques important and useful, this work is usually

considered an item of lower priority compared to the development of the communication procedures and protocols of the systems. Therefore only few individuals and groups are actively involved in this work. In order to follow up the contributions made by this study, we foresee the following points for further study:

(a) In collaboration with the interested parties within Working Group 1 of ISO/TC 97/SC 6, to elaborate a complete, formal specificatiom of the HDLC link layer protocol (balanced and unbalanced case).

(b) To apply the principles and description methods proposed in the Appendices 2, 3 and 4 to other existing protocol standards, and new systems under development. In particular:

(b1) Formal (and precise) description of the Virtual Circuit "end-to-end" service.

(b2) Precise descriptions of the services provided by the transport, session and presentation control layers of the Open Systems Architecture (ISO) or Public Data Network Service Applications (CCITT).

(b3) Application of formal protocol description techniques during the design of new protocol standards.

(c) To develop verification tools for system designs that use the proposed formal description methods and language.

(d) To develop methods for verifying that a real system, which implements a standard layered architecture, abides to the rules of the given communication standards.

REFERENCES

[1] G.V. Bochmann, "Specification and verification of computer communication protocols", submitted to Computer Networks.

[2] IEEE standard 488-1975; see also D.E. Knoblock et al., "Insight into interfacing", IEEE Spectrum (May 1975).

[3] CCITT, Recommendation X.25 (1976).

[4] ISO/TC 97/SC 6 N 1543 (Canada), "Formalized specification of HDLC".

[5] ISO/TC 97/SC 6 N 1569 (Germany), "State diagrams - definition of control modules (building blocks) of a data station".

[6] G.V. Bochmann, "Notes on the X.25 procedures for virtual call establishment and clearing", ACM Comp. Comm. Review 7, 4 (Oct. 1977) 53-59.

[7] CCITT, SG VII, contribution D37 to April 1978 meeting (IFIP), "Technical improvements to CCITT Recommendation X.75".

[8] G.V. Bochmann, "On the definition of the service interface of a protocol", INWG (IFIP WG 6.1), Note #170, 1978.

[9] G.V. Bochmann and F. Vogt, "Message Link Protocol-functional specifications", ACM Computer Comm. Review 9, 2 (April 1979), pp. 7-39.

APPENDIX 1

ISO
INTERNATIONAL ORGANIZATION FOR STANDARDIZATION
ORGANISATION INTERNATIONALE DE NORMALISATION
TC97/SC6

DATA COMMUNICATION: PROJECT 16

Source: Canada

Title: Comments on state diagram descriptions

1. Introduction and conclusions

Proposals for developing a state diagram description of the HDLC elements and classes of procedures were reported in documents N1569 and N1543. The present paper develops further certain issues which were brought up in the mentioned documents. The main conclusions are the following:

- For ease of understanding, the protocol description should be structured into several modules.

- An exact definition of the communication service provided by the protocol should be developed (possibly in collaboration with SC16). The provision of the service should be explicitly shown by the state diagram description.

- The mentioned documents agree in many points. There are differences in the format (but not meaning) of the description language, in the functional interfaces between the modules, and in the amount of detail to be included in the description.

- The state diagram description should have a one-to-one correspondence with the HDLC standard.

2. Modular structure

2.1 The state diagram description should be structured into a number of modules, with the result that an HDLC station is built out of a certain number of such modules, some of which are optional. The following modules are identified:

1. Response mode control module (determines response mode and P/F bit control). The following module types are distinguished:

   (a) Primary

      (a1) two-way alternate mode

      (a2) two-way simultaneous mode

   (b) Secondary in NRM

   (c) Secondary in ARM

2. Link set-up and disconnection module. The following module types are distinguished:

   (a) Unbalanced primary

   (b) Unbalanced secondary

   (c) Balanced

3. Information sending module. Depending on the complexity of the check-pointing operation, the following module types may be distinguished:

   (a) General

   (b) NO-REJ (only applicable with a No-REJ type of information receiving module in the opposite station).

   (c) Simple (only applicable with a response mode control module of type (a1) or (b) in the same station).

4. Information receiving module. The following module types are distinguished:

   (a) No-REJ

   (b) With-REJ

   etc.

5. Optional modules, such as Identification, One-way-Reset, etc.

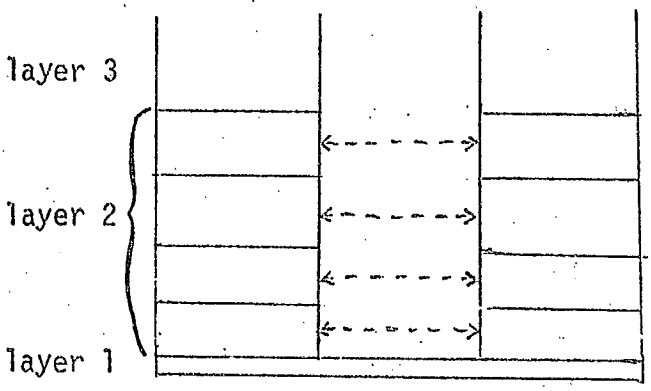
2.2 A typical HDLC station contains

- a response mode control module,
- a link set-up and disconnection module, and
- an information sending or receiving module (or both) depending on the need.

For example, an HDLC station following the balanced class of procedures (DP 6256) would contain modules of the types 1 (a2), 1 (c), 2 (c), 3 (a), and 4 (b).

3. Link layer service definition

3.1 ISO/TC97/SC16 is developing a layered architecture for Open Systems (see for example SC16 N117 or SC6 N1727) in which the HDLC classes of procedures are typical protocols for the link layer (layer 2). The part of the link protocol to be described by state diagrams (i.e. the rules for sending commands and responses) represents only a sublayer within the link layer, as shown in the figure below.



3.2 It is important to give an exact specification of the communication service provided by the link layer to the layer above. The service is provided through the upper layer interface. Different forms of interfaces may be adopted in different systems. Therefore the definition of the service should be, as much as possible, independent of the particular interface.

3.3 As an example, Annex 1 gives a definition of the service provided by an HDLC protocol. The definition uses a set of abstract "service primitives". A service primitive is an element of the provided service, making abstraction from the particular interface. A service primitive may be invoked (i.e. its execution may be initiated) by either side, the link layer or the layer above. It may provide for the exchange of parameter values.

3.4 The way how the service is provided by the protocol may be defined by including, in the state diagram description of the protocol, the execution of the service primitives. As an example, Annex 2 shows how the service primitives defined in Annex 1 could be included in the description of the link set-up and disconnection module given in N1543.

4. Module interfaces

The following module interfaces must be considered:

4.1 The layer interface through which the communication service is provided to the next higher layer (see section 3).

4.2 The interface to the next lower sublayer of the link layer for sending and receiving commands and responses.

4.3 Inter-module functional interfaces must be defined in the state diagram description.

4.4 The interface with the link manager is implementation dependent (not specified by the HDLC standard).

5. Comparison of state diagram description languages

5.1 The description languages used in N1569 and N1543 are based on similar concepts: The state of a module is defined by the place of a "token" in a diagram and the values of certain variables. When certain conditions are satisfied, transitions may be triggered, which involves the execution of an action and changes the state of the module.

5.2 In both cases, a rudimentary high-level programming language is used to describe the conditions and actions that relate to the variables.

5.3 N1543 uses a simple state diagram plus a table containing the definition of conditions and actions for describing the transitions, whereas N1569 uses annotated state diagrams, including the definition of conditions and actions in the diagram. Both methods are equivalent.

5.4 The functional interfaces between the modules of a station (point 4.3 above) are described differently in the two documents.

6. Equivalence between the state diagram description and HDLC standard specifications

6.1 It is proposed, that a state diagram protocol description should be developed which corresponds to the HDLC standard specification. It should also consider the operations at the layer interface through which the communication service is provided to the "user".

6.2 In addition, a more detailed description could also be useful as an implementation guide.

6.3 The protocol description of document N1569 contains many details which may be part of an implementation guide, but which are not specified in the standard.

Annex 1: Service provided by an HDLC protocol

1. List of service primitives (at the layer interface of a given station)

↓ SM : Set Mode primitive initiated by the entity using the service (in layer above)

↑ SM : Set Mode primitive initiated by the HDLC station

↓ uns. SM : Unsuccessful Set Mode primitive

↑ DISC : disconnection initiated by ...

↓ DISC : disconnection initiated by ...

↓ Send (data): primitive for sending a data block

↑ Receive (data): primitive for receiving a data block

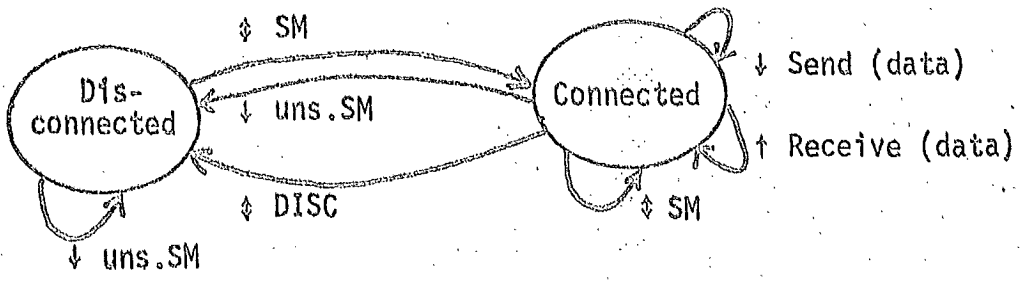
Status functions

- circuit-inoperable : true..false (becomes true after "too many" retransmissions)
- outstanding : 0..7
- not-yet-sent : integer

Notes: (a) The arrows "↑" and "↓" indicate which layer initiates the primitive. "↕" means "↑" or "↓".
(b) The status functions do not influence the operation.
(c) The " Send" and " Receive" primitives are provided by the information sending and receiving modules, respectively. The other primitives are provided by the link set-up and disconnection module

2. Local rules for using the primitives

The possible orders of execution for these primitives at a given station are defined by the transition diagram below. The data parameter of the Send or Receive primitives is arbitrary, provided its length is not too long ($\leq \ell$ max). The status functions may be called any time (between the execution of primitives).



Note: This diagram represents an abstraction of the operation of the HDLC protocol at the given station.

3. Global properties of the service primitives

(a) For each (successful) Set Mode primitive executed at the end of the link where it is initiated, there is at least one execution of such a primitive at the same time at the other end. (This is not in general true for the DISC primitive; for example, in the case of a circuit failure, a primary station may execute the DISC primitive without the secondary noticing).

(b) The sequence of data parameters passed by the Receive primitives between two consecutive Set Mode executions is identical to the sequence of the first data parameters passed by the Send primitives at the opposite side of the link between two corresponding Set Mode executions.

(c) Refering to (b) above, if n_r and n_s are the numbers of Receive and Send executions, respectively, then $(n_s - n_r)$ is ≥ 0, and lies between not-yet-sent and (not-yet-sent + outstanding). (i.e. $(n_s - n_r)$ data parameters (data blocks) are lost).

Annex 2: Service primitives in the protocol definition

The service primitives defined in Annex 1 may be included in the definition of the link set-up and disconnection module, given in section 5 of the Annex of N1543, by making the following changes to the protocol description:

(a) For the primary station:

(a1) Include "↓SM-request" as additional condition (enabling predicate) of the SXRM transition.

(a2) Include "↓DISC-request" as additional condition of the DISC transition.

(a3) Include "↓SM-confirm" as additional action of the UA transition starting in the Wait-for-SXRM-ack state.

(a4) Include "↓DISC-confirm" as additional action of the UA transition starting in the Wait-for-DISC-ack state.

(a5) Include an additional transition from the Wait-for-SXRM-ack state to the Disconnected state, with the enabling predicate "circuit inoperable" (which is implementation dependent) and the action "SM-failed".

(b) For the secondary:

(b1) Include "↑SM" as additional action of the SXRM transition.

(b2) Include "↑DISC" as additional action of the DISC transition.

Notes:

(1) The "↓SM" (or "↓DISC") service primitive is realized by the succession of the signals "↓SM-request" and "↓SM-confirm" (or "↓DISC-request" and "↓DISC-confirm", respectively). Similarly "↓uns.SM" is realized by "↓SM-request" followed by "↓SM-failed".

(2) The possibility that the secondary station requests a disconnection by sending a DM frame is not included in the description of N1543. Therefore, in this case, the "↓DISC" primitive is only executed in the primary station, and the "↑DISC" primitive only in the secondary.

APPENDIX 2

LM79-2.2

Question: New/VII

Original: English

Date: January 1979

SOURCE: Department of Communications: Canada

TITLE: Some principles for the specification of communication services and protocols.

1. Introduction

1.1 This paper was prepared by a Canadian expert at the request of the Federal Department of Communications. The purpose of this paper is to expose some principles which should be followed when writing specifications for communication services and protocols. The paper shows a direction for developing a general method of writing formal service and protocol specifications, which should contribute to the improvement of the quality of descriptions of standard specifications. This paper concentrates on general principles and the specification of layer services. A companion paper ("Methods for exact protocol specifications") considers methods for formal protocol specifications.

1.2 The paper relates to points 8 and 9 of Attachment I of Annex I of COMVII No... (Report of the First Rapporteur Meeting on Layered Models...), to point 2 of Annex II, and points 1 and 5 of Annex IX.

1.3 As examples of formal specifications, the Appendix contains a specification of the service provided by the X.25 LAP B link layer, and a specification of the LAP B link set-up and disconnection procedure is given in the companion paper.

2. Conclusions

2.1. The architectural model for PDN services should be developed to include exact specifications of the services provided by each of the identified layers.

2.2. The definition of a layer service includes three parts: (a) the list of abstract "service primitives", (b) local rules for using the service, and (c) global ("end-to-end") properties.

2.3. The definition of the service provided by a given layer is an abstraction of the layer and the layers below. It is the logical basis for the operation of the layers above, and serves as reference for the validation of the protocol.

3. Specification of communication standards

3.1 What should the description of a communication system layer include?

The following elements should be defined for each standard communication system layer within the architectural model of PDN services and applications:

(a) Relations between entities, connections, etc. which exist within the layer and their relationship to such objects in the adjacent layers. In particular, this part deals with multiplexing.

(b) The service provided by the layer.

(c) The service required from the layer below.

(d) The protocol followed by the entities of the layer.

The protocol is usually defined by giving specifications for the entities of the layer. This includes the specifications of

(i) the format of the data-units exchanged between the entities through the layer below, and

(ii) rules determining the order in which these data units may be exchanged in order to provide the service of the layer.

3.2. General principles

3.2.1. The remaining part of this paper concentrates on methods for formal specifications. Formal specifications of layers and entities seem to be necessary for obtaining workable, exact and non-ambiguous communication standards. However, formal specifications should be complemented with informal descriptions of why's and how's, written in natural language with diagrams, examples, etc.

3.2.2. It is probably necessary to dispose of several levels of descriptions ranging from high level (probably incomplete and ambiguous) to detailed, complete and unambiguous levels (probably not easily legible).

4. Exact service definitions

The following considerations relate to points (b) and (c) of section 3.1.

4.1. Abstract service primitives

4.1.1. The service of a layer is provided through the upper layer interface. Different forms of interfaces (for the same service) may be adopted in different parts of a distributed system. Therefore the definition of the service should be, as much as possible, independent of the particular interface through which it is provided.

4.1.2. A possible method is to define a particular service by a set of abstract "service primitives". A service primitive is an element of the provided service, making abstraction from the particular interface. A service primitive may be invoked (i.e. its execution may be initiated) by either side, service providing and using layers. It may provide for the exchange of parameter values.

4.1.3. As an example, the Appendix contains a list of service primitives for the service provided by the X.25 LAP B layer to the X.25 packet layer.

4.2. Order of execution

4.2.1. Usually, the service primitives that may be executed by a given entity may not be executed in an arbitrary order and with arbitrary parameter values. The permissible execution orders and parameter values must be defined. This involves (a) local rules, and (b) global "end-to-end" properties. The global properties are essential for defining the communication service.

4.2.2. These considerations are illustrated by the service provided by the X.25 LAP B layer (see Appendix). A local rule , for example, states that the DCE must execute successfully the Set Mode primitive before it may execute a Send primitive for sending a packet over the link. Global properties, for example, state that the successful execution of a Set Mode primitive by the DCE is always accompained by a simultaneous execution of such a primitive by the DTE, and that the next Receive primitive at the DTE delivers the same packet to the packet layer which was provided as parameter for the execution of the Send primitive at the DCE.

5. Specification of protocol standards

The following considerations relate to point (d) of section 3.1.

5.1. A formal specification should be a generative definition, i.e. such that all possible interaction sequences may be generated from the definition. This is in contrast to time-space sequence diagrams and similar methods which are useful for showing certain features of a protocol, but only define some possible interaction sequences.

5.2. Only those aspects of the operation of the entity should be defined which are required for obtaining compatibility with the peer entities. Clearly, additional aspects of the operation could be described, but these aspects should rather be called "possible implementation choices" or "implementation guide" (and not be part of the standard).

5.3. A general approach to formal, generative protocol description is explained in a companion paper ("Methods for exact protocol specification").

5.4.1. We note that the specifications for a communicating entity, which include specifications for the interaction with the entities in the layer above, contain more details than the definitions of the service primitives and their local rules of execution at the service interface.

5.4.2. For example, the local rules for using the service of the X.25 LAP B layer, given in section 3 of the Appendix, do not contain all the details of the LAP B protocol, as defined in X.25. They are much simpler, and easily derived from the specifications of the LAP B entity, given in the companion paper.

5.4.3. The protocol of a layer may be validated by showing that the global properties of the service provided can be deduced from the operation of communicating entities and the properties of the service provided by the layer below.

5.4.4. Therefore, the specification of the service of a given layer is an abstraction of the protocols which are executed in the given layer and the layers below.

APPENDIX

Service provided by the LAP B layer of X.25

1. List of service primitives (at a local service interface):

↓ SM : Set Mode primitive initiated by the entity using the service (in layer above) (only in DTE)
↑ SM : Set Mode primitive initiated by the entity of the layer in question (only in DCE)
↓ uns. SM : unsuccessful Set Mode primitive (only in DTE)
↑ DISC : disconnection initiated by ... (in DTE and DCE)
↓ DISC : disconnection initiated by ... (only in DTE)
↓ Send (data): primitive for sending a data block
↑ Receive (data): primitive for receiving a data block

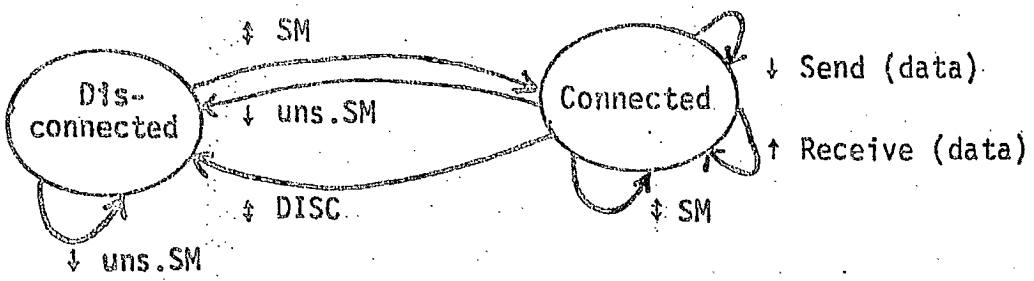
Status functions:

- circuit-inoperable: true..false (becomes true after "too many" retransmissions)
- outstanding : 0..7
- not-yet-sent: integer

Notes: (a) The arrows "↑" and "↓" indicate which layer initiates the primitive. "↕" means "↑" or "↓".
(b) The status functions do not influence the operation.
(c) The " Send" and " Receive" primitives are provided by the information sending and receiving modules, respectively. The other primitives are provided by the link set-up and disconnection module

2. Local rules for using the primitives

The possible orders of execution for these primitives at a given station are defined by the transition diagram below. The data parameter of the Send or Receive primitives is arbitrary, provided its length is not too long ($\leq \ell$ max). The status functions may be called any time (between the execution of primitives).



3. Global properties of the service primitives

(a) For each (successful) Set Mode primitive executed at the end of the link where it is initiated, there is at least one execution of such a primitive at the same time at the other end. (This is not in general true for the DISC primitive; for example, in the case of a circuit failure, a primary station may execute the DISC primitive without the secondary noticing).

(b) The sequence of data parameters passed by the Receive primitives between two consecutive Set Mode executions is identical to the sequence of the first data parameters passed by the Send primitives at the opposite side of the link between two corresponding Set Mode executions.

(c) Refering to (b) above, if n_r and n_s are the numbers of Receive and Send executions, respectively, then $(n_s - n_r)$ is ≥ 0, and lies between not-yet-sent and (not-yet-sent + outstanding). (I.e. $(n_s - n_r)$ data parameters are lost).

APPENDIX 3

LM79-3.2

Question: New/VII

Original: English

Date: January 1979

SOURCE: Department Of Communications: Canada

TITLE: Methods For Exact Protocol Specifications

1. Introduction

1.1 This paper was prepared by a Canadian expert at the request of the Federal Department of Communications. In view of the large number of different protocols which are and will be developed for PDN services, methods for exactly specifying these protocols should be available. Difficulties with using natural language protocol descriptions suggest to complement plain language descriptions with exact formal specifications.

1.2 A general approach to formal protocol descriptions is presented, which is based on the concept of "states and transitions". The purpose of this paper is to point out the need for a formal protocol specification method, and to suggest a general direction for developing such a method.

2. Conclusions

2.1 Methods for formally specifying protocols exist. They give rise to concise and exact definitions which are relatively understandable.

2.2 Formal specifications should be elaborated for the new protocols

of the different layers of the Layered Model, and should be considered as complements to the protocol descriptions in plain language.

2.3. Formal specifications of existing protocol standards should be elaborated and could be added to the recommendations in the form of an annex. (The Appendix contains a partial specification of the X.25 LAP B procedures).

2.4. Agreement on a method and language for formal protocol specifications is needed. For this purpose, the approach shown in the Appendix is proposed.

3. Framework for formal specifications

This paper considers a layered system architecture, where the protocol description is given in terms of a specification for the interacting entities (see also the companion paper "Some principles for the specification of communication services and protocols").

3.1 A formal specification should be a generative definition, i.e. such that all possible interaction sequences may be generated from the definition. This is in contrast to time-space sequence diagrams and similar methods which are useful for showing certain features of a protocol, but only define some possible interaction sequences.

3.2 To make the specifications of a given entity more simple and understandable, it is often advantageous to consider an entity to be built out of several modules; controlling different "sublayers", each performing separate "functions".

3.3 An entity (or one of its modules) is defined by its possible states and transitions between these states. The possible states are defined by
(a) a transition diagram (containing a finite number of "places"), or
(b) a set of variables (the "local variables" of the entity), each of which may assume a certain set of values, or both, (a) and (b).

3.4. At each instant in time, the entity either is in one of the possible states (defined by the places containing a "token" and/or the values of the local variables), or executes a transition, in which case the state is undefined.

3.5. A transition is defined by (a) a condition which must be satisfied before the transition may be started, and (b) an action

which is executed during the transition. The condition may depend on
(a1) the placement of tokens in the transition diagram*,
(a2) the values of local variables, and
(a3) received "signals" through the interaction with other local entities or modules.

The execution of an action may involve
(b1) a new distribution of tokens in the transition diagram*,
(b2) new values assigned to local variables, and
(b3) the sending of "signals" to other local entities or modules.

3.6. An example, defining the LAP B link set-up and disconnection procedure of X.25, is shown in the Appendix.

4. Local interaction between entities or components

This section is concerned with the interaction between entities of two adjacent layers through the layer interface, as well as interaction between entities or modules within the same layer through a functional interface.

4.1. The execution of a single "service primitive" (see definition in the companion paper), which, in an abstract form, describes the provision of an element of the service to an entity in the layer above, may appear to the entities involved as several distinct "signals" .
For example, a Set Mode primitive, initiated by the layer above, is realized by the succession of the SM request and SM indication signals (see Appendix).

4.2. Several other schemes may be useful for defining formally the local interaction of system modules, such as for instance
- direct coupling [see ref. 1, 2] ,
- hierarchical coupling [see ref. 2,3] ,
- reading or writing access to local variables of other entities or modules,
- state linkage [see ref. 4] ,

5. Possible representations of formal specifications

5.1. Specifications such as decribed in section 3 may be represented

---

* We leave for further study whether only one token per diagram is allowed (transition diagram of "finite state" type) or whether the number of tokens may vary (as in the case of Petri nets).

in many different, but equivalent ways. To simplify the reading of such specifications, it is important to adopt an appropriate representation in which the specifications are written. An important objective is to obtain (as much as possible) concise and easily understandable specifications.

5.2. The following are some examples of representations that have been used for specifications in a framework similar to the one of section 3:

5.2.1. Use of a high-level programming language, for instance specifying a transition by "when <condition> do <action>".

5.2.2. Naming transitions in a graphical diagram and defining the transitions in a table, using elements of a programming language for specifying the conditions and actions [see for example Appendix, and ref. 2] .

5.2.3. Defining, within the diagram, the conditions and actions of the transitions. [See ref. 4, 5; in ref. 5 actions are written into a box to distinguish them from conditions] .

5.2.4. Use of flow charts. This approach was adopted for the SDL ("Specification and Description Language" defined by CCITT, see ref. 6) which may be adapted to the framework described in section 3.

REFERENCES

1. G.V. Bochmann, "Finite state description of communication protocols", to be published in Computer Networks.

2. Canadian contribution to ISO, TC97, SC6, "Formalized specification of HDLC".

3. G.V. Bochmann and T. Joachim, "Development and structure of an X.25 implementation", Publ. #292, Departement d'I.R.O., Université de Montréal, submitted to IEEE Trans. on SE.

4. C.A. Vissers, "Interface, a dispersed architecture", Proc. 3rd Arm. Symp. on Computer Architecture, Jan. 1976, Florida, pp. 98-104.

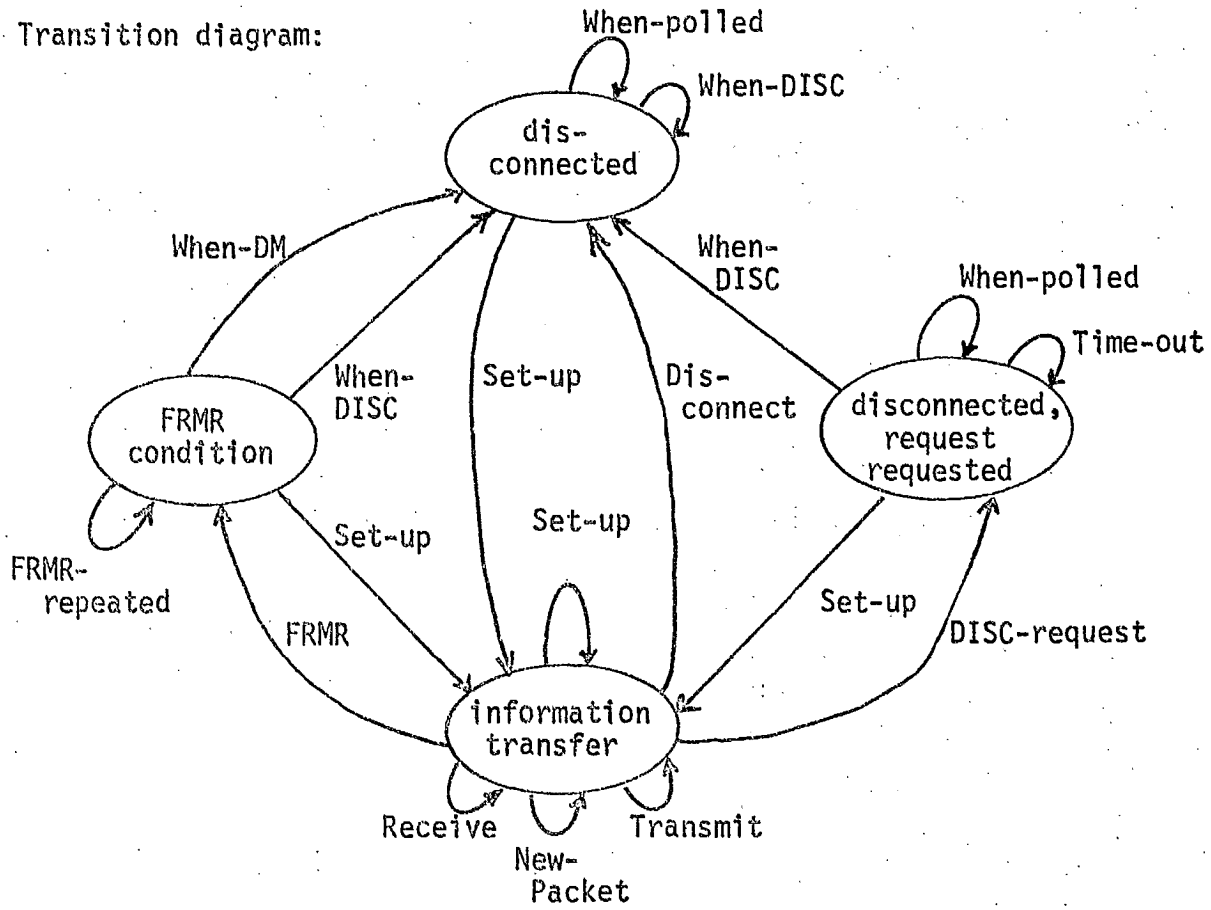
5. ISO/TC07/SC6 N 1569 (Germany), "State diagrams - definition of control modules (building blocks) of a data station".

6. CCITT, Rec. Z101-Z103, "Functional specification and description language (SDL)".

APPENDIX

Formal specification of LAP B link establishment and clearing procedure executed by the DCE

Transition diagram:



Local variables:

Count:	0..N2	(retransmission count)	
V(S) :	0..7	(send state variable)	} Only needed
V(R) :	0..7	(receive state variable)	} during the
V(B) :	0..7	(buffer state variable)	} information
Unack:	0..7	(last unacknowledged sequence number	} transfer phase

Definition of transitions

Notation : _X_ stands for reception or transmission of an "X" frame depending whether "_X_" represents a condition or action, respectively. (We note that the transmission error detection, frame format, frame delimitation and bit stuffing are handled by sublayers of the LAP B which are not described here). ¯X¯ stands for the execution of the service primitive "X" (at the layer interface above). The following service primitives are considered:

¯↑SM¯ : link set-up or reset

¯↑DISC¯: link disconnection

¯↓SEND(data)¯: sending a packet (data)

¯↑RECEIVE(data)¯: receiving a packet (data)

Name	Condition	Action	Reference to pertinent section of X.25
Set-up	_SABM_ {received}	↑SM; _UA_; {send UA} V(S) = 0; V(R) = 0; V(B) = 0 (reset buffer pointer); stop timer	2.4.5.1
Disconnect	_DISC_ {received}	↑DISC; _UA_; stop timer	2.4.5.3
When-DISC	_DISC_ {received}	_DM_ {send DM}	2.4.5.4.1
When-polled	received poll bit = 1	_DM_	
FRMR	...(see X.25, section 2.4.10)	_FRMR_ {send FRMR }	2.4.9.4
When-DM	_DM_ {received}	↑DISC	2.4.9.4
FRMR-repeated	received command except DISC, or SABM	_FRMR_	2.4.9.4
DISC-request	...(see X.25, section 2.4.10.2)	↑DISC; _DM_; start timer; count = N2	2.4.9.3
Time-out	Time-out and count ≠ 0	_DM_; start timer; count = count - 1	2.4.5.4.2
Some transitions during the information phase			
New packet	↓SEND(data) and V(B) ≠ Unack	place data parameter into V(B)th buffer; V(B) = V(B) + 1	reception of new data from the layer above

Transmit	V(S) ≠ V(B)
	and
	V(S) < Unack + modulus -2
		_I(N(S) = V(S); N(R) = V(R);
		information = V(S)th buffer)_;
		V(S) = V(S) + 1
			transmission of an I frame

Receive	_I(N(S);N(R);inf)_ and N(S) = V(R)	↑RECEIVE (data = inf); V(R) = V(R) + 1; Unack = N(R)	reception of the next expected I frame

APPENDIX 4

ISO
INTERNATIONAL ORGANIZATION FOR STANDARDIZATION
ORGANISATION INTERNATIONALE DE NORMALISATION
TC97/SC16

Source: Canada

Title: Comments on formal description techniques

1. Introduction and conclusions

This paper contains comments on Annex E of N 117 on formal description techniques for specifying the Open Systems Architecture. The sections 2 through 4 of this paper apply to the sections E2 through E4, respectively, of the annex of N117. Most of the paragraphs of these sections should be included in the text of the Annex E to the Reference Model.

2. Need for service definitions

2.1 The definitions of the services provided by the different layers of an Open System are an essential part of the specification of the Open Systems Architecture. The architectural model remains vague as long as the logical characteristics of the layer services are not clearly specified.

2.2 Therefore, the precise definition of the Architecture should include the parts listed below. This could be stated in a section of Annex E of the Reference Model under the title "Elements constituting a description of a computer communication architecture". (The topic of section E.2 of Annex E of N 117 deals essentially with the first part below).

2.3 The following parts should be defined for each standard Open System layer:
(1) Relations between entities, connections, etc. which exist within the layer and their relationship to such objects in the adjacent layers.

(2) The service provided by the layer.

(3) The service required from the layer below.

(4) The protocol followed by the entities of the layer.
The protocol is usually defined by giving specifications for the entities of the layer. This includes the specifications of
(4a) the format of the data-units exchanged between the entities through the layer below, and
(4b) rules determining the order in which these data units may be exchanged in order to provide the service of the layer.

3. Defining a layer service

3.1 The service of a layer is provided through the upper layer interface. Different forms of interfaces (for the same service) may be adopted in different parts of a distributed system. Therefore the definition of the service should be, as much as possible, independent of the particular interface through which it is provided.

3.2 A possible method for specifying a layer service is based on abstract "service primitives". A service primitive is an element of the provided service, making abstraction from the particular interface. A service primitive may be invoked (i.e. its execution may be initiated) by either side, service providing and using layers. It may provide for the exchange of parameter values. For specifying a particular service, a set of service primitives must be defined.

3.3 For certain considerations, it is not necessary to distinguish whether the service primitive is initiated by the entity using the service or the entity providing it. (For example, a "confirmed call request" and an "accepted incoming call", in X.25, give rise to the same connection). This should be supported by the notation for service primitives (see for example Annex 1).

3.4 Usually, the service primitives that may be executed by a given entity may not be executed in an arbitrary order and with arbitrary parameter values. The permissible execution orders and parameter values must be defined. This involves (a) local rules, and (b) global "end-to-end" properties. The global properties are an essential part of the communication service definition.

3.5 These considerations are illustrated by the example of Annex 1 which gives a possible definition of the link layer service. The definition is structured into three parts:
(a) list of service primitives (*Initialize, Terminate, Send, Receive*),
(b) local rules,
(c) global properties.
A local rule, for example, states that an entity using the service must execute successfully the Initialize primitive before it may execute a Send primitive for sending a data unit over the link. Global properties, for example, state that the successful execution of a Initialize primitive by one entity is always accompained by a simultaneous execution of such a primitive by the peer entity, and that the next Receive primitive executed by the latter delivers the same data unit which was provided as parameter for the execution of the Send primitive by the former.

4. Specification of protocol standards

4.1 General

4.1.1 A formal specification of a protocol should be a generative definition, i.e. such that all possible interaction sequences may be generated from the definition. This is in contrast to time-space sequence diagrams and similar methods which are useful for showing certain features of a protocol, but only define some possible interaction sequences.

4.1.2 Only those aspects of the operation of the entity should be defined which are required for obtaining compatibility with the peer entities. Clearly, additional aspects of the operation could be described, but these aspects should rather be called "possible implementation choices" or "implementation guide" (and not be part of the standard).

4.1.3 To make the specifications of a given entity more simple and understandable, it is often advantageous to consider an entity to be built out of several modules; controlling different "sublayers", each performing separate "functions".

4.2 Use of protocol specifications

4.2.1 For protocol verification during the design: The correct operation of the protocol of a layer may be verified by showing that the global properties of the service provided can be deduced from the operation of communicating entities in the layer and the properties of the service provided by the layer below.

4.2.2 As a guide for designing real implementations: Of course, it is not forbidden to follow the specification of a protocol standard and additional aspects of operation (see point 4.1.2 above) as a design example. In fact, if the architecture is properly done, achieving some "good" partitioning of function from the layered structure concept, one may recommend such a practice. Nevertheless, a number of technical differences between a protocol description and an actual product, such as stemming from requirements for different levels of parallelism, can be forseen. How close an actual implementation will be to the corresponding protocol description remains an implementation choice.

4.2.3 For verifying that a real system abides to a standard architecture: There seem to be three approaches to this verification, namely

(a) To verify that the system behaves in accordance with the specifications of a proper set of protocols.

(b) To verify that the system functions correctly in a context where it is connected with other systems that abide the Open Systems Architecture and protocols.

(c) To record a log of the service primitives executed during the operation of the system, and to check that this log could have been generated by a system abiding the Open Systems Architecture and protocols. (Note: This approach seems to be easier to automate than the other approaches. However, only the execution sequences occuring during some particular testing will be verified).

4.3 Methods for formal protocol specifications

Agreement on a method to be used for the formal specification of the protocols used in the Open System Architecture is desirable. The following paragraphs present a possible approach to such a method.

4.3.1 Description of entities

4.3.1.1 The protocol of a layer is specified by defining certain rules for the behaviour of the entities in the layer. The behaviour of an entity may be described by the following model of states and transitions.

4.3.1.2 An entity is defined by its possible states, and transitions between these states. The possible states are defined by
(a) a transition diagram (containing a finite number of "places"), or

(b) a set of variables (the "local variables" of the entity), each of which may assume a certain set of values, or both, (a)and (b).

4.3.1.3 At each instant in time, the entity either is in one of the possible states (defined by the places containing a "token" and/or the values of the local variables), or executes a transition, in which case the state is undefined.

4.3.1.4 A transition is defined by (a) a condition which must be satisfied before the transition may be started, and (b) an action which is executed during the transition. The condition may depend on
(a1) the placement of tokens in the transition diagram*,
(a2) the values of local variables, and
(a3) received "signals" through the interaction with other local entities or modules.

The execution of an action may involve
(b1) a new distribution of tokens in the transition diagram*,
(b2) new values assigned to local variables, and
(b3) the sending of "signals" to other local entities or modules.

4.3.1.5 An example, defining the LAP B link set-up and disconnection procedure of X.25, is shown in the Annex 2.

4.3.2 Local interaction between entities

This section is concerned with the interaction between entities of two adjacent layers through the layer interface, as well as interaction between entities (or modules) within the same layer through a functional interface.

4.3.2.1 The execution of a single "service primitive" (see section 3.2), which, in an abstract form, described the provision of an element of the service to an entity in the layer above, may appear to the entities involved as several distinct "signals". For example, a Initialize primitive, initiated by the layer above, is realized by the succession of the Init request and Init indication signals (see Annex 2).

4.3.2.2 Several other schemes may be useful for defining formally the local interaction of system modules, such as for instance
- direct coupling [see ref. 1, 2] ,
- hierarchical coupling [see ref. 2,3] ,
- reading or writing access to local variables of other entities or modules,
- state linkage [see ref. 4] .

*We leave for further study whether only one token per diagram is allowed (transition diagram of "finite state" type) or whether the number of tokens may vary (as in the case of Petri nets).

4.3.3 Possible representations of formal specifications

4.3.3.1 Specifications such as described in section 4.3.1 may be represented in many different, but equivalent ways. To simplify the reading of such specifications, it is important to adopt an appropriate representation in which the specifications are written. An important objective is to obtain (as much as possible) concise and easily understandable specifications.

4.3.3.2 The following are some examples of representations that have been used for specifications in a framework similar to the one of section 4.3.1:

(a) Use of a high-level programming language, for instance specifying a transition by "when <condition> do <action>".

(b) Naming transitions in a graphical diagram and defining the transitions in a table, using elements of a programming language for specifying the conditions and actions [see for example Annex 2, and ref. 2] .

(c) Defining, within the diagram, the conditions and actions of the transitions. [See ref. 4, 5; in ref. 5 actions are written into a box to distinguish them from conditions] .

(d) Use of flow charts. This approach was adopted for the SDL ("Specification and Description Language" defined by CCITT, see ref. 6) which may be adapted to the framework described in section 4.3.1.

REFERENCES

1. G.V. Bochmann, "Finite state description of communication protocols", Computer Networks 2 (1978) 361-372.

2. ISO/TC97/SC6N1543 (Canada), "Formalized specification of HDLC".

3. G.V. Bochmann and T. Joachim, "Development and structure of an X.25 implementation", Publ. #292, Departement d'I.R.O., Université de Montréal, submitted to IEEE Trans. on SE.

4. C.A. Vissers, "Interface, a dispersed architecture", Proc. 3rd Arm. Symp. on Computer Architecture, Jan. 1976, Florida, pp. 98-104.

5. ISO/TC07/SC6 N 1569 (Germany), "State diagrams - definition of control modules (building blocks) of a data station".

6. CCITT, Rec. Z101-Z103, "Functional specification and description language (SDL)".

Annex 1: Service provided by an HDLC protocol (example of a link layer service)

1. List of service primitives (at the layer interface of a given station)

↓Init: Initialize primitive initiated by the entity using the service (in layer above)

↑Init: Initialize primitive initiated by the HDLC station (entity of the link layer)

↓ uns. Init: Unsuccessful Initialize primitive

↑ Term : Termination initiated by ...

↓ Term : Termination initiated by ...

↓ Send (data): primitive for sending a service data unit

↑ Receive (data): primitive for receiving a service data unit

Status functions

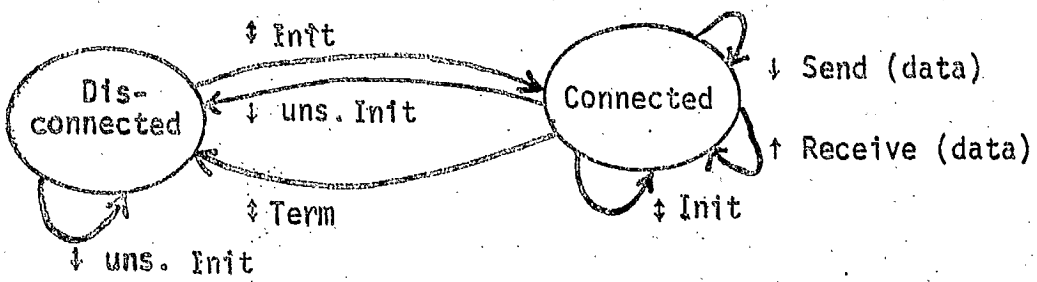
- circuit-inoperable : true..false (becomes true after "too many" retransmissions)
- outstanding : 0..7
- not-yet-sent : integer

Notes: (a) The arrows "↓" and "↑" indicate which layer initiates the primitive, i.e. the entity below or above the service interface, respectively. "↕" means "↑" or "↓"

(b) The status functions do not influence the operation.

2. Local rules for using the primitives

The possible orders of execution for these primitives at a given station are defined by the transition diagram below. The data parameter of the Send or Receive primitives is arbitrary, provided its length is not too long ($\leq \ell$ max). The status functions may be called any time (between the execution of primitives).



Note: This diagram represents an abstraction of the operation of the HDLC protocol at the given station (operation of the link layer protocol), as described in Annex 2.

3. Global properties of the service primitives

(a) For each (successful) Initialize primitive executed at the end of the link where it is initiated, there is at least one execution of such a primitive at the same time at the other end. (This is not in general true for the Terminate primitive; for example, in the case of a circuit failure, the entity using the service at the primary station may execute the Terminate primitive without the secondary noticing).

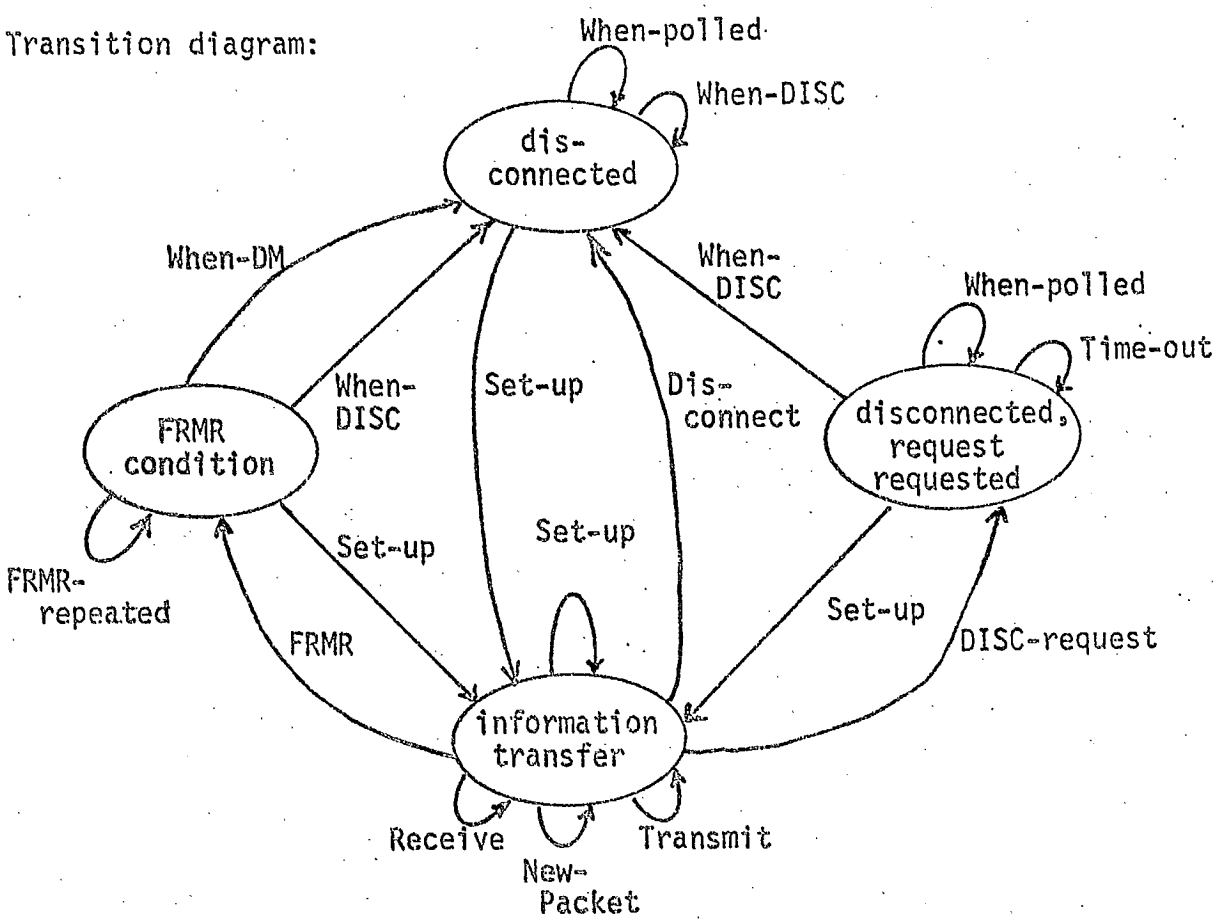
(b) The sequence of data parameters passed by the Receive primitives between two consecutive Initialize executions is identical to the sequence of the first data parameters passed by the Send primitives at the opposite end of the link between two corresponding Initialize executions.

(c) Refering to (b) above, if n_r and n_s are the numbers of Receive and Send executions, respectively, then $(n_s - n_r)$ is ≥ 0, and lies between not-yet-sent and (not-yet-sent + outstanding). (I.e. $(n_s - n_r)$ data units are lost).

Annex 2

Formal specification of LAP B link establishment and clearing procedure executed by the DCE

Transition diagram:



Local variables:

Count:	0..N2	(retransmission count)	
V(S) :	0..7	(send state variable)	} Only needed
V(R) :	0..7	(receive state variable)	} during the
V(B) :	0..7	(buffer state variable)	} information
Unack:	0..7	(last unacknowledged sequence number	} transfer phase

Definition of transitions

Notation : _X_ stands for reception or transmission of an "X" frame depending whether "_X_" represents a condition or action, respectively. (We note that the transmission error detection, frame format, frame delimitation and bit stuffing are handled by sublayers of the LAP B which are not described here). ¯X¯ stands for the execution of the service primitive "X" (at the layer interface above). The following service primitives are considered:

¯↑Init¯ : link set-up or reset

¯↑Term¯ : link disconnection

¯↓SEND(data)¯: sending a packet (data)

¯↑RECEIVE(data)¯: receiving a packet (data)

Name	Condition	Action	Reference to pertinent section of X.25
Set-up	_SABM_ {received}	↑Init; _UA_; {send UA} V(S) = 0; V(R) = 0; V(B) = 0 (reset buffer pointer); stop timer	2.4.5.1
Disconnect	_DISC_ {received}	↑Term; _UA_; stop timer	2.4.5.3
When-DISC	_DISC_ {received}	_DM_ {send DM}	2.4.5.4.1
When-polled	received poll bit = 1	_DM_	
FRMR	...(see X.25, section 2.4.10)	_FRMR_ {send FRMR }	2.4.9.4
When-DM	_DM_ {received}	↑Term	2.4.9.4
FRMR-repeated	received command except DISC, or SABM	_FRMR_	2.4.9.4
DISC-request	...(see X.25, section 2.4.10.2)	↑Term; _DM_; start timer; count = N2	2.4.9.3
Time-out	Time-out and count ≠ 0	_DM_; start timer; count = count - 1	2.4.5.4.2
Some transitions during the information phase			
New packet	↓SEND(data) and V(B) ≠ Unack	place data parameter into V(B)th buffer; V(B) = V(B) + 1	reception of new data from the layer above

Transmit	V(S) ≠ V(B)
	and
	V(S) < Unack + modulus -2
		_I(N(S) = V(S); N(R) = V(R);
		information = V(S)th buffer)_;
		V(S) = V(S) + 1
			transmission of an I frame

Receive	_I(N(S);N(R);inf)_ and N(S) = V(R)	↑RECEIVE (data = inf); V(R) = V(R) + 1; Unack = N(R)	reception of the next expected I frame