Zeitschrift: Bulletin des Schweizerischen Elektrotechnischen Vereins, des Verbandes Schweizerischer Elektrizitätsunternehmen = Bulletin de l'Association suisse des électriciens, de l'Association des entreprises électriques suisses

Herausgeber: Schweizerischer Elektrotechnischer Verein ; Verband Schweizerischer Elektrizitätsunternehmen

Band: 73 (1982)

Heft: 5

Artikel: Structuring of Communication Modern Systems

Autor: Vissers, C. A.

DOI: https://doi.org/10.5169/seals-904941

Structuring of Communication in Modern Systems

C. A. Vissers

621.39

Starting with the example of the general purpose instrumentation interface IEEE-488, which was developed in a period where only small and medium scale integrated components were available, it is shown which new insights in user requirements, modern technology, and system design methodology have lead to a modern approach in system structuring and hardware and software design.

Vom Beispiel des IEEE-488 Interface für allgemeine Anwendung ausgehend, das in einer Zeit entwickelt wurde, wo nur Komponenten geringer und mittlerer Integrationsdichte verfügbar waren, wird gezeigt, welche neuen Erkenntnisse der Anwenderbedürfnisse, moderner Technologie und Methodik der Systementwicklung zu einem modernen Zugang zur Systemstrukturierung und zum Entwurf von Hardware und Software geführt haben.

Partant de l'exemple de l'interface IEEE-488 qui fut développé à une époche où seuls des composants à faible et moyenne integration étaient disponibles, l'auteur montre les nouveaux aspects des besoins des utilisateurs, de la technologie moderne et de la méthodologie de la conception des systèmes qui ont conduit à une approche moderne de la structuration des systèmes et de la conception du matériel et du logiciel.

1. The significance of standardization

In May 1974 Working Group 3 of the Technical Committee 66 of the International Electrotechnical Commission (IEC TC 66, WG 3) pretty well completed a draft standard for an Interface System which later would be known as IEC-625, IEEE-488, GPIB, HP-IB, etc. Today, it is hardly thinkable that a device meant for laboratory automation or production test facilities would not be equipped with this Interface. Even equipment such as computers and peripheral devices which were not in the primary scope of this standardization process, frequently offer this Interface as an option.

By mid 1981, this standard has been translated into 9 different languages, and over 300 companies offer over 3000 different instruments, controllers, systems and components which use, or are based upon this Interface. These numbers seem to double every two years, and it may well be expected that this standard will serve us till up in the nineties, rather than up in the eighties, which was the originally expected lifetime.

This paper was presented at the SEV-meeting of 8th October 1981 on 'normalisation dans le domaine des systèmes micro-informatiques'.

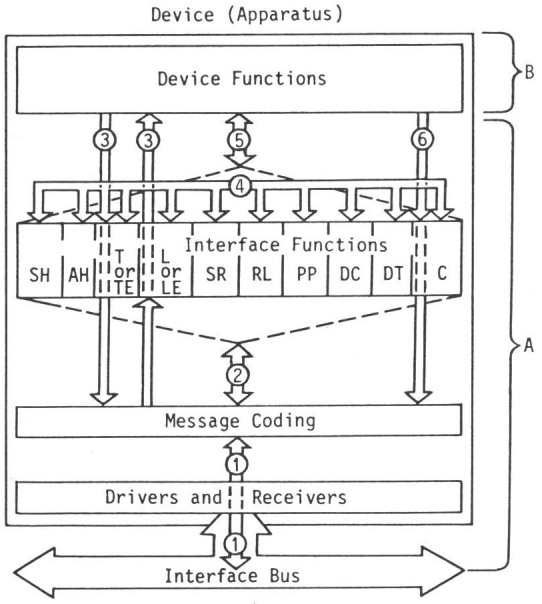

Fig. 1 Structure of a device from the perspective of the IEC-625 Interface system for programmable measuring apparatus
Important is the distinction into layers of Interface functions, Message coding, Drivers and Receivers, and the Interface bus. Each interface function is represented by a formal specification in terms of a finite state machine

In analysing the 'why' and 'how' of this success, there are at least three key factors that can be distinguished:

1. The timing was right: Whereas sometimes standardization committees are filibustered to unproductivity, this standard was initiated while there was a great need for it, and produced by WG 3 within two years.

2. The technical quality was right: The technical concepts in this standard allow a wide variety of simple and complex instruments to interwork in an efficient way, and thus to bridge the gap between the period that devices were assembled from discrete (or at most medium scale integrated) components to nowadays, where technology allows us to install more and more functions in a single device.

3. The specification was right: Whereas interface specifications are notorious sources of system failures, as provoked by hidden ambiguities in the usually informal specification, the highly structured (figure 1) and formal specification of this interface guarantee unambiguous interpretation by designers and users, and thus real compatibility between devices.

These aspects of structuring and formality, in particular how they are seen now by the international community, form the main subject of this text.

2. Reference Model

A mechanism to cope with the enormous complexity of protocol (another term for interface) structures in modern distributed systems, in which each device or subsystem can be another computer system, is by the use of so called 'Reference Models'. In this context, the ISO 'Reference Model for Open Systems Interconnection (OSI)', which was developed by ISO TC 97 SC 16, plays a dominant role at this moment. As it may be expected that every system designer, who incorporates a standard interface or protocol in his system, sooner or later will get confronted with this model, we base our discussion on it.

The objectives of the OSI Reference Model is to provide the *framework* for the development of *standard* interface, protocol, and service specifications. These standards will allow users of computer systems that are interconnected via local area and/or public and/or satellite (computer) networks, to communicate and to share each others facilities (i.e. their systems are 'open'). Many applications are served with this

development, the economic interest is great, as illustrated by the fact that far over hundred top-experts out of the world's highest industrialized countries cooperate in SC 16.

The following shows a strongly simplified image of the Reference Model. This image is used to introduce the definitions of Interface, Protocol, and Service, which have a confined semantics in the context of this model:

The Reference Model employs a vertical layering of (open) systems (S1...Ss), and a horizontal layering of functional entities (E1...Ee) within each system (fig. 2). On this layered structure a number of concepts are defined, which, in increasing order of abstraction, are called: Interface, Protocol, and Service (fig. 3).

An *Interface* is the interaction between two adjacent but in different layers located functional entities within one system, e.g. the Ee/Ee-1 interaction within S1.

A *Protocol* is the interaction between two functional entities located in the same layer, but pertaining to different systems, e.g. the Ee-1/Ee-1 interaction of S2 and Ss. This interaction is not directly but proceeds via the underlaying Service, in this example the Ee-2 Service.

A *Service* is the integration of all functional entities below a certain layer, e.g. the Ee-3 Service is the functional integration of all functions (i.e. all Ee-3's through El's including the communication media for interaction between S1...Ss) below the Ee-2 layer, and represents the behaviour of these functions as seen by all Ee-2's.

These abstract concepts are handled according to a great number of technical criteria to develop practical interface, protocol, and service designs. This is further discussed in section 3.

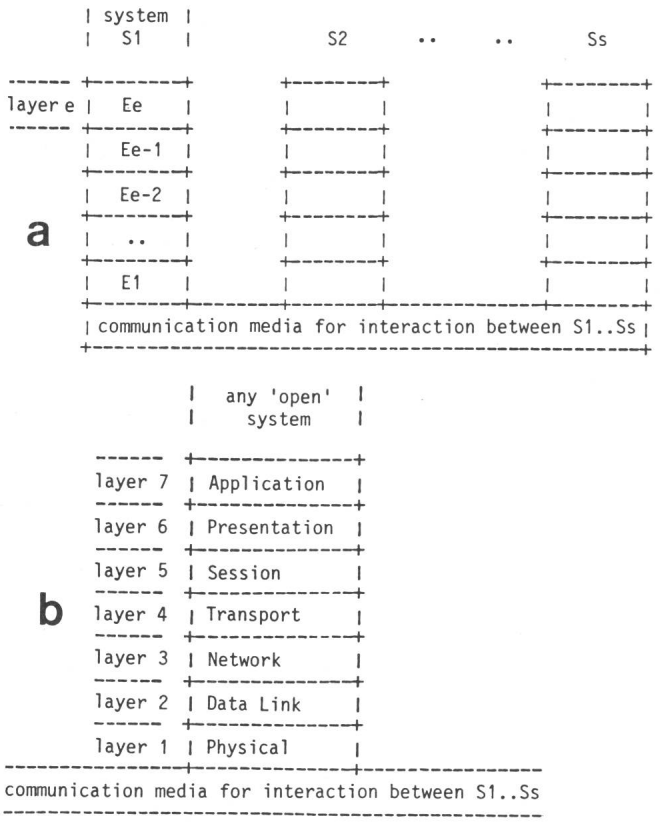

Fig. 2 The ISO «OSI Reference Model» for open systems interconnection

Figure a: The model uses 7 layers, which from top to bottom are as shown in figure 2b

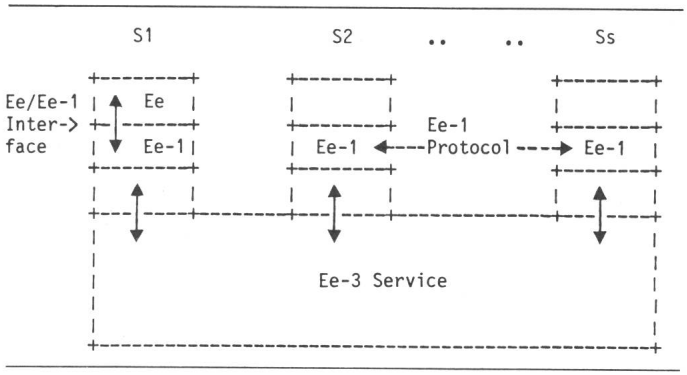

Fig. 3 Graphical representation of the Interface, Protocol, and Service concepts

In the following, an attempt is made to draw a rough correspondence between the IEC-625 structure and the OSI Reference Model. This correspondence, of course, cannot be complete and precise.

The Device (Apparatus) of figure 1 is an example of a system in figure 2, which is open in the sense of being accessible by other devices which use the same IEC-625 standard. The boxes in figure 1 (the IEC-625 'Reference Model') are examples of (functions within) entities of figure 2, which may be located within a layer of figure 2b as follows:

- the Interface Bus is an example of a communication medium,
- the Drivers and Receivers are within the Physical layer,
- the Device Functions are within the Application layer,
- the Interface Functions and Message Coding have to be distributed among the layers.

This distribution can be performed in the following way: First of all the message coding should be devided into two sections. One section concerns the representation of application data. A separate document, the IEC-625 'Code and Format Conventions', is dedicated to this subject. This document corresponds to the Presentation layer. The dotted lines through the Talker (T) and Listener (L) functions, indicating the transparency of data transferred through these functions, allow this interpretation of figure 1. The other section concerns the representation of protocol data, and should be associated with the interface functions to which these data apply.

The Remote Local (RL), Parallel Poll (PP), Device Clear (DC), and Device Trigger (DT) functions are typical examples of 'standardizable portions' of application functions, and as such belong to the Application layer. The Source Handshake (SH) and Acceptor Handshake (AH) are functions that control the valid transfer of data, and belong to the Data Link layer. The Talker (T) and Listener (L) functions are used for the setting up of a connection between devices and can be located in the Transport layer. The Service Request (SR) and parts of the Controller (C) functions have to do with the setting-up and synchronization of connections, and may be located at Session level.

The IEC-625 does not know a network function.

3. Role of Interface, Protocol, and Service concepts

The definitions shown in figure 3 allow a number of conceptual manipulations which are useful to illustrate the roles of these definitions in the development of open systems standards. In the following discussion we use the Transport Service as an example.

3.1 The Service concept

From the definition of the service concept it will be clear that the functionality of the transport service, which is represented by figure 4a in the most direct and simple way, can also be represented by the compound representation of figure 4b. This compound representation consists of the Transport Entities (TEs), which embody the Transport Protocol, on top of the Network Service.

It follows from these different representations that the same transport service can be rendered by different network services on top of which different transport protocols are placed. Thus, a transport service can be a standard, although the underlying mechanisms may be non-standard. This allows the development of standard protocols on top of the transport service, as well as a step-by-step replacement of lower level non-standard protocols by standard protocols without making higher level protocols obsolete.

Service specifications, therefore, represent a vital mechanism in distributed systems to balance and adapt in a standard way the requirements of many different applications to the characteristics rendered by many different network technologies. An important criterium in this process is that on each level in the model a service is defined, such that each higher level service screens off ('wraps') user-unfriendly characteristics of the next lower level service.

It follows also that the transport service defines the ultimate constraints for the development of the transport protocols and the network service and as such permits a top-down approach in the development of protocols, starting out with service user requirements.

Yet, service specifications are 'just discovered' and service standards for practical applications are still in an early stage of development and dedicated to a limited goal: the connection oriented services (see 5.2). The OSI 'Transport Service for Connection Oriented Services', whose draft proposal may be expected by mid 1982, will most possibly be the first Service standard in the world.

3.2 The Protocol concept

The transport protocol, which is the interaction between two or sometimes more transport entities, determines the functional specification of the transport entities. Since the functional specification of an open system is determined by the functional specification of all its entities, it follows that the protocols of all layers determine the implementation of an open system.

Protocols are always based on a lower level service. Even the physical protocol is based on the (usually rather poor) service of the underlying communication medium. This service, however, is not further decomposed in another protocol and service, but implemented directly by the communication medium.

Entities exchange specific messages, called 'Protocol Data Units' (PDUs), to perform the protocol and to provide the required service. In the OSI model PDUs are exchanged as data, and thus transparently, via the underlying service.

This is shown in figure 5 for the Transport Protocol Data Units (TPDUs).

Protocols make use of such mechanisms as cyclic redundancy check, retransmission, time-out, buffering, numbering of PDUs, window-mechanism, etc. Sometimes the same mechanisms can be found at different levels in the model.

3.3 The Interface concept

From figure 4 and figure 6 it will be clear that the interactions between the transport service and the session entities (fig. 6a) must be the same as the interaction between the transport entities and the session entities (fig. 6b), as a necessary (but not sufficient) requirement to conclude that the integrated and distributed representation of the transport service are equivalent.

These interactions are modelled by the transport interfaces. As such the transport interfaces form a kind of lock, which can be used to map a transport protocol on the transport service.

The transport interface, as meant in this way, is actually an abstract interface between abstract representations of transport service, transport entities, and session entities. This is because the OSI standards do not want to prescribe the precise way how the implementation of the transport and session entities exchange information. These decisions concern the *implementation* interface, which is the domain of the manufacturer.

It follows, that when a system is implemented, the abstract interfaces need to be implemented as well as the abstract entities. It follows therefore, that any hardware and/or software implementation interface may be chosen between two adjacent entities, whereas any hardware and/or software implementation may be chosen for any entity.

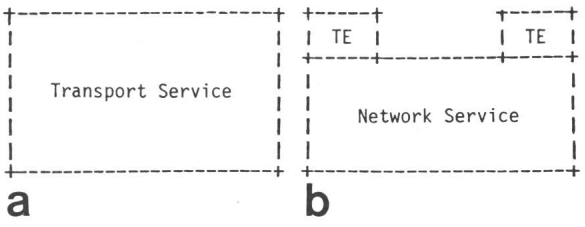

Fig. 4 Integrated (6a) and distributed (6b) representation of the Transport Service

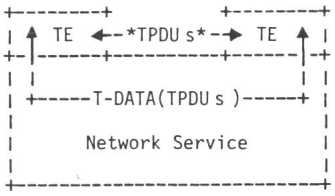

Fig. 5 The virtual (*TPDUs*) and the real T-DATA(TPDUs) exchange of Transport Protocol Data Units between Transport Entities

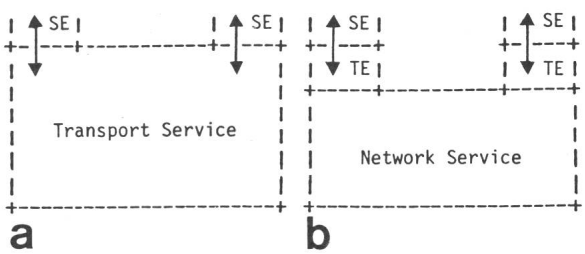

Fig. 6 Integrated (a) and distributed (b) representation of the Transport Service, showing two Transport Interfaces

Abstract interfaces are, like service specifications, modelled by temporal orderings of 'Service Primitives', which shall be discussed in the next section. These service primitives play also an essential role in protocol specifications in addition to the PDUs. In the remaining part of this paper, therefore, we restrict ourselves to Service specifications.

4. Modelling of Service specifications

Obviously, adequate specification techniques are required to reflect the right level of abstraction in interface, protocol, and service specifications. Conventionally, specifications are written (informally) in plain language, and illustrated with drawings, diagrams, and tables. In recent years the virtue of formal specification techniques has become more and more appreciated as a tool during the design process, and as a means of communication between the original designers and the users.

A formal specification (because of the formal basis) avoids the ambiguity of the informal specification, and thus cannot be explained in different ways. It allows also the logical analysis of a specification, such that deadlocks, races, instabilities, etc. can be detected in an early phase during the design process. These analysis procedures can be, and sometimes are, automated on a computer. Formal specifications can also be concise and clear, and as such they support the human reasoning during the design process.

The informal specification language, however, remains indispensable as a meta language to define the formal language, as a means to explain and introduce formal specifications, and as a means for intuitive communication during the design process.

Therefore, formal and informal language complement, rather than compete, each other! In order to perform this complementary role, it is necessary that both formal and informal language use the same semantics for the concepts that they use in common.

In the following we introduce some concepts which appear to be highly valuable in the specification of interfaces and services.

4.1 Service Primitives

A Service Primitive (SP) is a *unit of interaction* between two adjacent entities, or between an entity and its adjacent service.

This unit of interaction can be conceived as an elementary process of finite duration with a specific semantics, expressed by the *type* of SP. Usually this semantics concerns the exchange of a set of particular data elements. These data elements are called parameters or parameter values.

Thus an SP is specified by its type and by the values of its parameters.

The concept of *interaction* implies that one does not want to distinguish how each entity contributes to this interaction. Instead, the interaction is interpreted so as to contribute to the activity of both entities in the same way.

The concept of *unit* of interaction implies that one does not want to give an inner temporal ordering structure (i.e. relevant to the parameters) to the SP.

Figure 7 tries to illustrate these concepts. In this figure the enclosed surface of entity X symbolically represents the activity of entity X. The little surface marked SP belongs to the activity of entity X as well as to the activity of entity Y.

4.2 Temporal Ordering

A Temporal Ordering (TO) defines the allowable time sequences of service primitives. These time sequences as well as references of parameter values to each other are used to indicate causality, which forms the basis of a specification.

A specification by way of the enumeration of all possible time sequences of SPs usually becomes very tedious to read. Therefore shorthand notations (temporal ordering primitives) are used in practice. An example is given in figure 8. The figures of section 5 show examples of temporal orderings by way of simple timing diagrams. The dotted lines in these diagrams indicate causality, the vertical lines indicate time.

5. Types of Services

A type of service defines a fixed pattern of service primitives in a temporal ordering specification. Types of services are used to characterize and satisfy the needs of particular service users. In this context the service user may be an end-user, but may also be an arbitrary entity in the open system, e.g. the session entity as the user of the transport service.

In the following we introduce a number of service types which are currently under discussion. For simplicity we eliminate the parameters, and focus on the transport service by way of example.

5.1 Connectionless Services

Service types are currently classified as connection and connectionless services. The connectionless service is the more simple class, so we start with this class.

5.1.1 Unconfirmed Service

The unconfirmed service is again the most simple one of the connectionless services. Sometimes this service is called the '(unreliable) datagram', or the 'send and pray' service.

Figure 9 shows an example for data transfer. A 'data request' SP (Dr) is executed at the border of the calling session entity (SE) and the transport service (TS), which is normally followed by the execution of a 'data indication' SP (Di) at the border of the transport service and the called session entity. The information, however, may get lost in the transport service, in which case the data indication is not executed.

Parameters in data request are: 'called session entity' and 'data'. The parameters in Di are 'calling session entity', and 'data', which, except for a certain residual error rate, are the same data as in Dr.

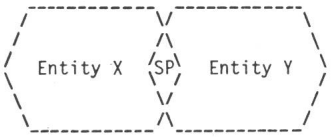

Fig. 7 Symbolic representation of the actions X and Y, and the interaction SP

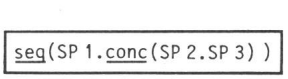

Fig. 8 Temporal Ordering of three SPs
It defines a sequence (seq) of SP 1 followed by interaction *x*, where *x* is a concurrent (conc) ordering of SP 2 and SP 3 (i.e. SP 2 and SP 3 may overlap in an arbitrary way)

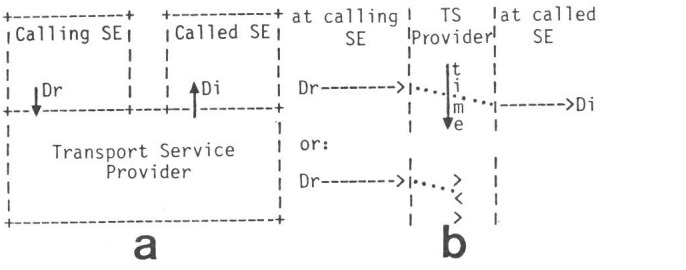

Fig. 9 Unconfirmed data Service
a Geographical distribution of SPs
b Two possible TOs of SPs

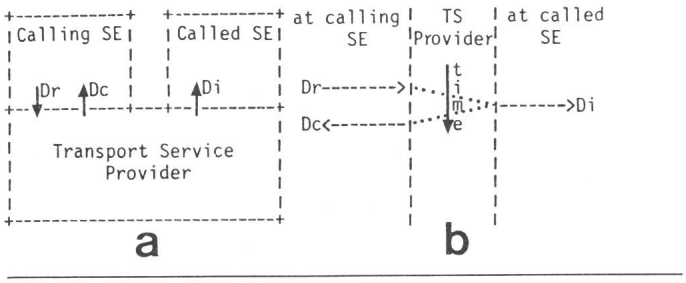

Fig. 10 The provider confirmed Service
a Geographical distribution of SPs
b The most probable TO of SPs in the provider confirmed Service

The unconfirmed service forms the basis of all communication: each transmission line performs an unconfirmed service. Several important applications require the unconfirmed service, for example teleconferencing (video-, or voice-). Also some forms of periodic status reporting and message broadcasting may use it.

5.1.2 Provider Confirmed Service

The provider confirmed service tries to avoid the uncertainty of the execution of the Di primitive by providing a 'data confirm' SP (Dc) or a 'confirmation of delivery' SP at the calling SE. This is shown in figure 10. Sometimes this type of service is called 'reliable datagram service'.

At protocol level, the transport entity can retransmit the Dr-PDU when a time-out condition indicates that the Dc-PDU has not arrived in time. This may be caused by the loss of the Dr-PDU, which implies that the Di-SP has not happened, or by the loss of the Dc-PDU, which implies that the Di-DP has happened but the confirmation by the receiving Transport entity is not received by the sending Transport entity.

It follows that this type of service is drastically more complex than the simple unconfirmed service as it implies all kinds of error procedures. Figure 10b therefore shows only the statistically most probable temporal ordering. The single parameter in the Dc-SP expresses either 'confirmed', or 'unconfirmed'.

Provider confirmed services are useful in some forms of process control applications, where absolute and timely guarantee of information delivery is necessary.

5.1.3 User Confirmed Service

In the user confirmed service, the user is requested to give an 'intelligent' response on the Di primitive by means of a 'data response' SP (Dp) (fig. 11). The parameter in the Dc-SP, therefore, has the same value as the parameter in the Dp-SP, or the value 'unconfirmed'.

The protocol for this type of service has almost the same complexity as the protocol for the provider confirmed service. User confirmed services have important applications, for example in banking, airline reservation, database inquiries, etc., where short predefined questions require short answers.

5.1.4 Relevance of connectionless services

Connectionless services have a particular relevance for Local Area Networks (LANs). Local area networks offer a number of inherent technological characteristics (such as short and relative constant transit delay, high bandwidth and broadcast transmission) in conjunction with relative low cost, that make these networks particularly suitable for a number of new applications as well as new and cheeper implementations of existing applications.

On the other hand the potential power of LANs can never be fully exploited independent of the long-haul terrestrial and satellite networks that provide the means to interconnect largely distributed LANs.

Service and protocol standards for LANs are now under development, in particular within the IEEE-P 802 group. These developments, however, are restricted to the bottom two layers of the ISO OSI-Reference-Model, whereas ISO TC 97 SC 16 is too much engaged in getting the connection oriented services and protocols ready for draft standards.

5.2 Connection Service

The connection service is the more complex type of service. It is required when service users interact on a regular basis (e.g. the interaction of a terminal with an application program) and/or with long units of data (e.g. in file transfer). The connection service is a very general service and thus can be used for many applications. It forms the basis of many public data communications networks which provide the X.25 Interface. This forms the background why ISO TC 97 SC 16 and SC 6 focus for the standards which are now under development on this type of service.

The connection service (fig. 12) employs three phases: establishment phase, data transfer phase, termination phase, in the cited sequence. The termination phase, however, may be invoked by either user after the 'connect' request or 'connect' indication primitive.

The establishment phase looks like a user confirmed service. Its purpose, however, is to set up a connection and to negotiate between both service users and the service provider under which conditions this connection will be maintained. These

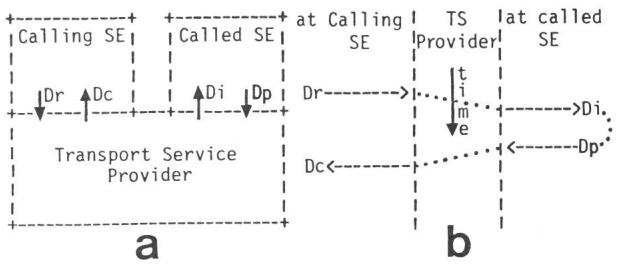

Fig. 11 The user confirmed Service
a Geographical distribution of SPs
b The must probable TO of SPs in the user confirmed Service

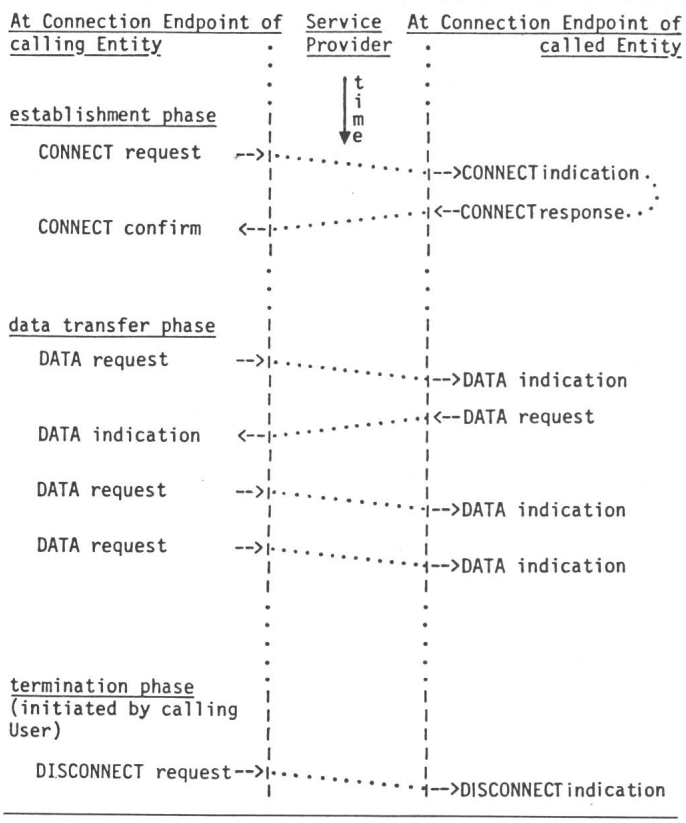

Fig. 12 Example of a Connection Service

conditions concern aspects of throughput, transferdelay, residual error rate, etc. and are usually called the quality of service.

In the data transfer phase 'normal data' can be transferred in either direction. This means that the connection can be conceived as a two way simultaneous (or: full duplex) virtual circuit between the calling and called user.

Data is offered to the service provider by means of the 'data request' SP, and delivered by the service provider by means of the 'data indication' SP. The service provider preserves the integrity of the data. The maximum length of the data is negotiated during connection establishment. The data is transferred 'transparently', i.e. any coding or formatting of the data is allowed.

Sometimes an additional full duplex virtual circuit is installed for short urgent messages. This circuit is operated by 'expedited data' request and indication primitives.

The termination phase eliminates the connection. This termination may be invoked by either user by issuing a 'disconnect request' SP. The service provider informs the other user by issuing a 'disconnect indication' SP. The Service provider can also terminate a connection if it cannot longer guarantee the quality of service agreed earlier. The service provider then informs both service users by way of the disconnect indicate SP.

Figure 12 shows an example. The temporal ordering presents only one possible sequence of SPs.

Bibliography

[1] *Lenzini, Popescu-Zeletin, Vissers:* State of the Art study on the standardization of level 4: Transport Layer of the ISO/TC97/SC16 Reference Model for Open Systems Interconnection – July 1981 – CEC publication.

[2] *Vytopil, Vissers, Karjoth, Steinacker, Rafiq:* Interaction Primitives in the Formal Specification of Distributed Systems – ISO TC97 SC16 WG1 FDT Washington-7 tutorial document – September 1981.

[3] An interface system for programmable measuring instruments (byte serial, bit parallel). IEC-Publication No. 625, 1979/80, IEEE Standard No. 488-1975.

[4] *Frank Chen:* Classes of Applications on the IEEE Standard Local Area Network – September 1980.

[5] *A. Lyman Chapin:* Connectionless Data Transmission – May 19, 1981 – prepared for X3 S33/X3 T56.

[6] *Allen B. Rochkind:* Considerations for selecting Types of Service required in a LNDLC standard; a view from the top down – September 1980.

[7] ISO DP7498: Draft Proposal – Reference Model for Open Systems Interconnection – November 1980.

[8] ISO TC97 SC16 N380: Formal Description Techniques group – Proposed Guidelines for the specification of Services, Protocols, and Interfaces (Part 2 – Working Paper) – (Amsterdam, June 23–27, 1980).

[9] ISO TC97 SC16 N381: Formal Description Techniques group – Proposed Guidelines for the specification of Services, Protocols, and Interfaces (Part 1) – (Chicago, January 28–30, 1980, and Amsterdam, June 23–27, 1980).

[10] ISO TC97 SC16 N697: Draft Connection Oriented Transport Service Specification – Berlin – June 1981.

[11] ISO TC97 SC16 N698: Draft Connection Oriented Transport Protocol Specification – Berlin – June 1981.

Author's address

Prof. Dr. ir. *Chris A. Vissers,* Twente University of Technology, Enschede NL.