NEAR EAST

1988

UNIVERSITY

Faculty of Engineering

Department of Computer Engineering

COM 400 Graduation Project

OPEN SYSTEMS

Submitted to : Prof. Dr. Fakhreddin Mamedov

Submit by : Cihat Boyacı 940722

CONTENTS

CHAPTER 1

INTRODUCTION

Introduction has always played a pivotal role in our society. The ability to gather, process, store and distribute information has been a key factor in the growth of most civilisations in the history of mankind. During the second half of the twentieth century we have seen major technological developments which have transformed all our traditional notions of handling information. Information gathering and distribution has been supported by communication technology, and distribution processing and storage by computer technology.

Communication technology, as well as computer technology, have sustained an exponential growth during the last few decades. Highly reliable, wide bandwidth communication links have come into existence, providing easy communication over long distances. Computers have steadily become smaller, cheaper and powerful, to the extent that a single user workstation on a desk today has more computational capabilities than that was available to large organisations only a few years ago. The most remarkable development, however, has been the ability to implement system in which computer and communication technologies are integrated. Today most large organisations, with many offices in geographically drive areas, have the capability of routinely obtaining current information stored in any of their computers. As our ability to gather, process, store and distribute information improves, our desire the implement more sophisticated information processing functions and applications involving multiple organisations grows even more rapidly.

In order to implement large communication systems it is essential that the systems make use of some information processing capabilities. On the other hand, the computer systems of today with multiple processing units have to be able to transfer information from one place the another within the system. The merger of communication technology and computer technology is proceeding systematically and rapidly.

In Table 1.1 we present a simple classification of the organisation of processors and the kind of networks. When the distance between processor is small and they are located on the same board, they require connections with very high bandwidth. Such connections are found within a computer system and are considered as computer networks. The communication technique used is tightly integrated into the design of processors and other components on the board. Even when we consider processors which are up to a few meters apart, the communication is among processing elements within a single system. As distances grow from 100 meters to a few kilometres the communication bandwidth requirements are usually from 10K to 10M bytes Per second. These types of networks are referred to as Local Area Networks (LANs). The term Metropolitan Area Network (MAN) is used to refer to networks which interconnect processor within a metropolitan area. In order to connect processors which are farther away Wide Area Networks (WANs) are used. When connecting a group of processors on a local area network, for example to another similar group, an interconnection between the two LANs is required. Such connections fall within the domain of internet working. *Internet working has been used to interconnect a large number of networks throughout the world.*

Table 1.1 Classification of Network

Inter-processor distance	Processor Location	Bandwidth Range	Example Network
0.1 m	Circuit board	1-10 G Bytes/sec	Data flow machine
1 m	System	10M – 1G Bytes/sec	System
10 m	Room	100K – 10M Bytes/sec	LAN
100 m	Building	10K – 10M Bytes/sec	LAN
1 km	Campus	10K – 10M Bytes/sec	LAN
10 km	City	10K – 10M Bytes/sec	MAN
100 km	Country	1K – 1M Bytes/sec	WAN
1000 km	Continent	1K – 100K Bytes/sec	Internet work
10000 km	Planet	1K – 100K Bytes/sec	Internet work

In the OSI-RM, each system is decomposed functionally into a set of subsystems and is represented pictorially in a vertical sequence. Vertically adjacent subsystems communicate through their common interfaces, while *peer* subsystems collectively from a layer in the architecture. Each layer provides a set of well-defined services to the layer above, by adding its own functions to the services provided by the layer below. The layers of the model are partitioned as follows:

- *1: Physical layer* Achieves the transmission of row data bits over a communication channel (medium).
- *2: Data link layer* Converts the row transmission facility into a line that appears free of frames and delimiting them. This layer may also include access control to the medium, error detection and correction.
- *3: Network layer* Performs the routing and switching of data between any two systems across multiple data links and sub-nets.
- *4: Transport layer* Operates on an end-to-end basis achieving the necessary quality of service for the exchange of data between two end systems. May include end-to-end error recovery and flow control.
- *5: Session layer* Allows users on different machines to establish sessions between them, and hence establishes and messages communication dialogue between processes.
- *6: Presentation layer* Manages and transforms the syntax of structured data being exchanged. Is also concerned with the semantics of the information transmitted.
- 7: *Application layer* Deals with the information exchange between end-system application processes and defines the messages that may be exchanged.

The above layering was created according to the original design principles used in the construction of the ISO model. According to this:

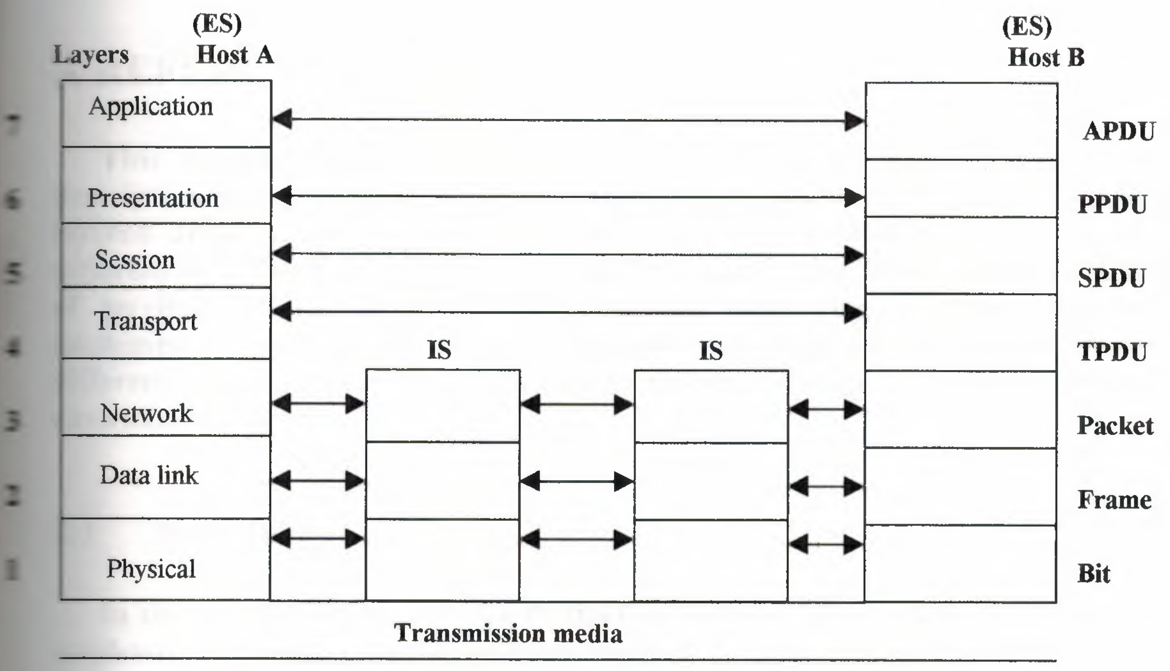


Figure 1.1 The OSI reference model.

1. **Different levels of abstraction are placed in separate layers.**
2. **Similar functions are grouped together within a single layer with each layer performing a well defined function.**
3. **The function of each layer is chosen so as to be amenable to the definition of a standard protocol.**
4. **Minimization of information flow across interfaces in a primary goal in drawing the layer boundaries.**

Although the majority of network architectures widely in use are based on the principles of layering, most do not fit the OSI model exactly in their allocation of layers and protocols used. Examples of these are the IBM's SNA (Meijer 1987), DECnet and DARPA Internet (Quarterman and Hoskins 1986), to name but a few. Conversely, some new network architectures, such as the MAP (General Motors 1988; O'Prey 1986) and TOP (Boeing 1988), have adopted the OSI-RM for their architecture and hence form 'open' networks. Open networks use internationally standardized procedures for communications rather than local or proprietary ones.

CHAPTER 2

A REFERENCE MODEL

This chapter is an introduction to Open Systems Interconnection and to its description at the highest level of abstraction. It includes a detailed discussion of its layered architecture. In particular we discuss the notations of services offered by different layers and protocols that govern communication within each layer. Discussion of services offered by each layer and the supporting protocols required to be implemented, however, are contained in subsequent chapters. A notation for identifying different objects, including data units, within the Open Systems Interconnection environment is given.

2.1 Introduction

In this section we present the distinction between *open system* and *real system,* and emphasize the point that the primary concern of Open System Interconnection is with the externally visible behavior of systems. It is pointed out that the Reference Model is simply an abstract model that permits a detailed specifications of interactions between open systems.

2.1.1 Open Systems

Open Systems Interconnection (OSI) is concerned with exchange of information between systems, in fact, between *open systems.* Within OSI, a distinction is made between real systems and open systems. A real system is a computer system together with the associated software, peripherals, terminals, human operators, physical processes, and even sub-systems that are responsible for information transfer. It is assumed that the components of a computer system listed above form an autonomous whole and are in themselves capable of processing and transferring information. On the other hand, an open system is only a representation of a real system that is known to comply with the architecture and protocols as defined by OSI. In fact, the representation takes into account only those aspects of a real system that pertain the information exchange between such open systems and are consistent with OSI. Put differently, an open system is that portion of a real system which is visible to other open systems in their attempts to transfer and process information jointly.

2.1.2 Systems Interconnection

Information transfer is not the only concern of OSI. The term systems connection suggest much more. It also includes aspects that are necessary for systems to work together co-operatively towards achieving a common, though distributed, goal. These aspects are:

1. *Inter-process communication*, which is concerned with information exchange and synchronisation of various activities undertaken by application processes.
2. *Data representation*, which is concerned with representation of information being exchanged, and with ways the define alternative representations for the variety of information;
3. *Data storage,* where the concern is with storage of data at possibly remote locations, and access to it;
4. *Process and resource management,* which concerns ways by which application processes are declared, initiated and controlled, and the means by which such application processes acquire resources available within the OSI environment;
5. *Integrity and security,* which concerns correctness and consistency of data and with access to data either during storage, exchange or processing;
6. *Program support,* which is concerned with providing an environment for program development and execution at remote locations.

While all six of the above activities have been identified to be immediate concern to OSI, the earlier emphasis was largely on information exchange and its representation. More recently, the concerns of OSI have shifted towards providing an environment wherein application processes co-operate by accessing computing resources and remote locations.

2.1.3 The Reference Model

Figure 2.1 provides and abstract model of OSI environment as it becomes available the application processes within open systems.

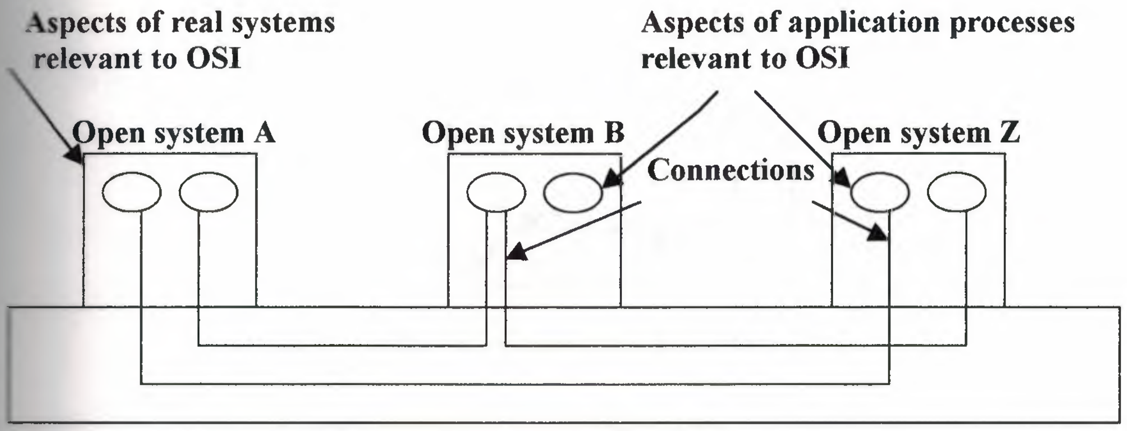

Aspects of real systems relevant to OSI
Aspects of application processes relevant to OSI
Open system A
Open system B
Open system Z
Connections

Figure 2.1 Model of the OSI environment.

Note that only open systems are considered within the model of the OSI environment, and within an open system only those portions of application processes that are relevant to OSI have been included in the model. Interaction between application processes takes place when they exchange information. The model, therefore, stipulates the need for a physical communication media of transmission of data. It is this abstract model which is elaborated upon by the Reference Model. In fact all of the international standards or recommendations within OSI provide varying degrees of detail about the functioning of open systems (or sub-systems) in this abstract model.

The Reference Model itself does not specify the external behaviour of open systems. It simply lays down the framework for a detailed specification of services and protocols to be supported by open systems. The major objective of the framework is to describe and crystallise the concept of layered architecture. Towards that, it is also provides a definition of certain key elements of OSI. In the light of the architecture developed and purposed seven layers, the Reference Model clarifies the notion of conformity of OSI standards. As such, Reference Model may be viewed as the highest level of (abstract) description of standards developed within OSI. The second level of OSI description is provided by a specifications of OSI services, and last, by OSI protocol specification. This relationship is illustrated in Figure 2.2 The Reference Model admits a large class of service specifications, only one which is shown in the figure. Similarly, a service specifications admits a large class of protocol specifications. Needless to say that a specification of services and protocols allows a variety of implementations.

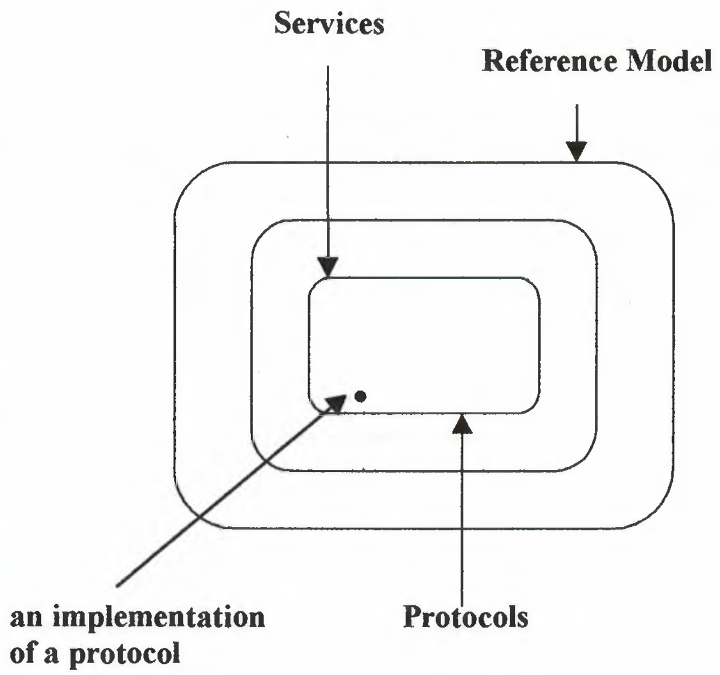


Figure 2.2: The relation between Reference Model, service specification and protocol specification.

2.2 The Layered Architecture

In this section we discuss the layered architecture of OSI, emphasising the point that this structure leads to a more modular approach, particularly from the viewpoint of developing standards and their implementation. The concepts of services, functions and protocols are discussed in some detail. Connections are also introduced in this section, but a more detailed discussion of connection-oriented data transfers is included in Section 2.5.

A network of interconnected systems may be viewed as just that. Such a view partitions network vertically into a number of distinct systems that are interconnected using a physical transmission media. The view presented earlier in Figure 2.1 is similar, expect that open systems are used to model real systems. This model views the network in its totality, but partitions it as a series of horizontal layers (see Figure 2.3). Here, a layer cuts across the vertical boundaries of systems. Such a view is helpful in more than one way:

- It allows for a discussion on exchange of information between peer objects within a layer, independent of other layers,
- It allows for gradual and modular development of functionally of each layer, and
- It is simultaneously allows open system to be viewed as a succession of sub-systems, thereby permitting a modular implementation of the open system itself.

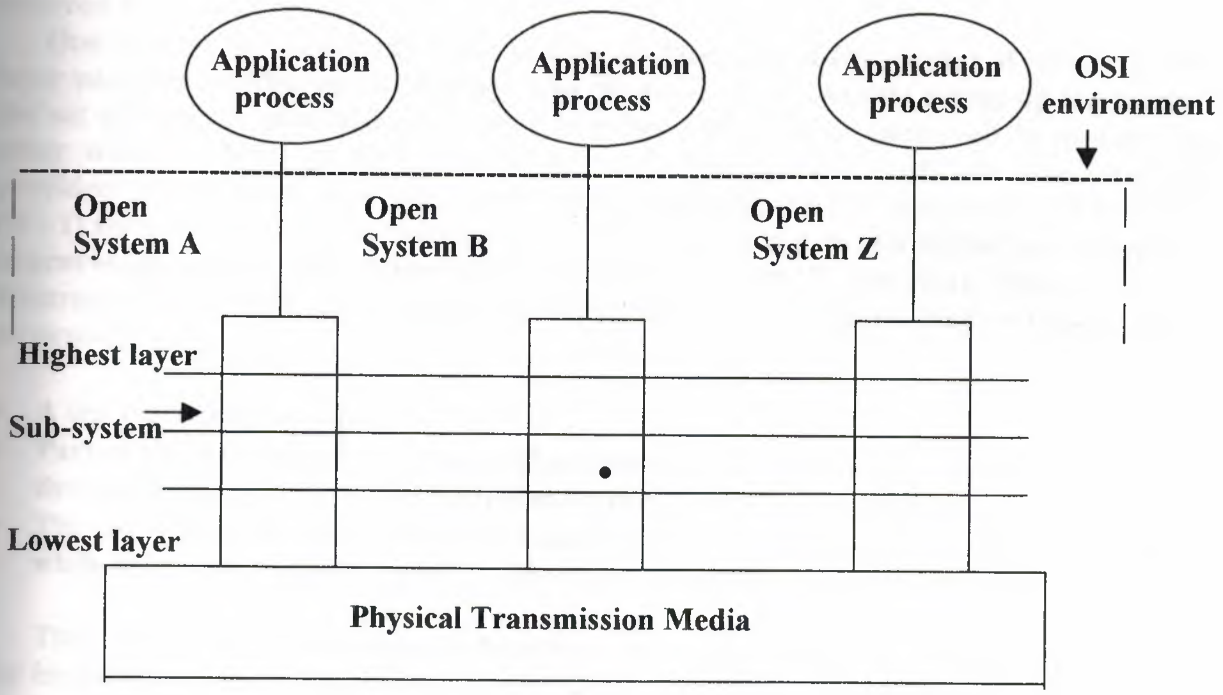


Figure 2.3 Layers and sub-systems in an OSI environment

2.2.1 Layers, Services and Functions

For simplicity, a given layer is referred to as an (N) layer, the one below it as (N – 1) layer and the one above as (N + 1) layer.

The succession of layers not only partitions the whole network, but it also partitions each system into a succession of sub-systems – a sub-system being identified (or formed) by the intersection of an open system and a layer. Sub-systems within a layer are said to be of the same rank, while sub-systems belonging to adjacent layers within an open system are said to be adjacent. Adjacent sub-systems communicate through their common boundary. Communication between sub-systems of the same rank is more complex. In fact, a major concern of OSI is to define the means to provide for such capability. Of course, communication between to adjacent sub-system is also subject to discussion and standardisation within OSI.

A sub-system is logically viewed as consisting of a number of entities. An entity is a representation of a process within a computer system. It is a software or hardware module which is active, or in some cases, a manual or physical process. It can take many forms, and is capable of autonomous actions by itself, or in response to requests or commands from other entities. In this regard, the notion of an entity is very similar to that of a process in a computer system. In OSI environment, however, the entities and their inter-relationship are well structured.

Note that only those aspects that are relevant to interactions within OSI are represented as part of entity. Thus, a layer may be viewed as consisting of a large number of entities that are spread across various open systems. At the highest layer entities model application processes while those below model software and hardware modules that are responsible for providing OSI services. At the bottom layer an entity allows access to the physical transmission media. An entity within an (N) layer is referred to as an (N) entity.

One concept that is central to the layered architecture of OSI is that of service. Each layer provides a different set of services to the layer above. As one moves up the layers, the set of services provided by a layer is either enhanced or improved in quality. In other words, a layer provide services to the layer above it, and also uses services provided by the lower layer, and those below. Basically, the (N) layer adds value to the (N – 1) services, and thereby enhances the set of services it provides to the layer above or improves upon them. This it does by implementing certain (N) functions. Figure 2.4 is an illustration of the layered architecture of OSI. There the hierarchy may be looked upon recursively as:

- A layer provides services,
- Part of the services are implemented as functions within the layer, while the rest are derived from (N – 1) services provided by the (N – 1) layer and those below;
- The (N – 1) services are partly implemented as (N – 1) functions in the (N – 1) layer, while others are derived from (N – 2) services, and those below. And so on.

Thus, the concept of layered architecture allows identification of different functions for implementation within various layers. This is, in fact, the usual top-down approach to designing systems. The functions to be implemented are specified in the form of services to be provided by each layer.

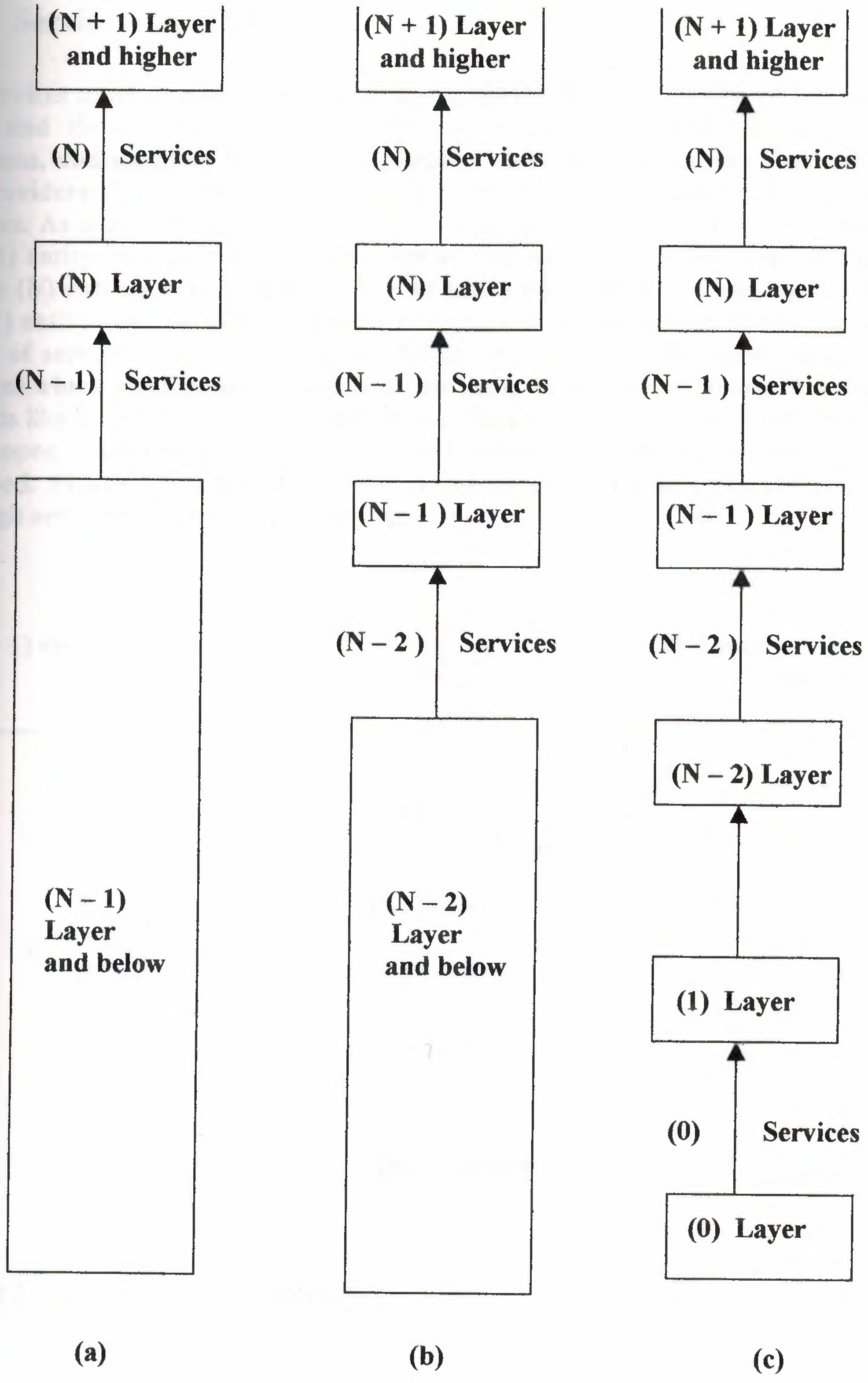


Figure 2.4 concept of layering in OSI architecture layers, services and functions.

2.2.2 Service Access Points

Services mace available by a layer are implemented in the form of functions in that layer and those below. Entities of the layer are responsible for implementing its functions, and similarly for the layers below. Thus, it is the entities that are ultimately the providers of services. Furthermore, it is the (N + 1) entities that are the users of (N) services. As a consequence, a service provided by an (N) layer may be accessed by an (N + 1) entity whenever it interacts with an (N) entity. There are however, restrictions on the (N) entities with which an (N + 1) entity may interact. First, the (N) entity and (N + 1) entity must be within the same open system. Further restrictions are specified in terms of service-access points. Formally, an (N) service-access-point, or (N)-SAP, is a point at which services are provided by an entity to an (N + 1) entity. A service-access-point is like an interface through which two entities from adjacent layers, but within the same open system, may interact with each other. In doing so, service is provided or accessed. Figure 2.5 is an illustration of the concept that services are accessible only through service-access points. Note that:

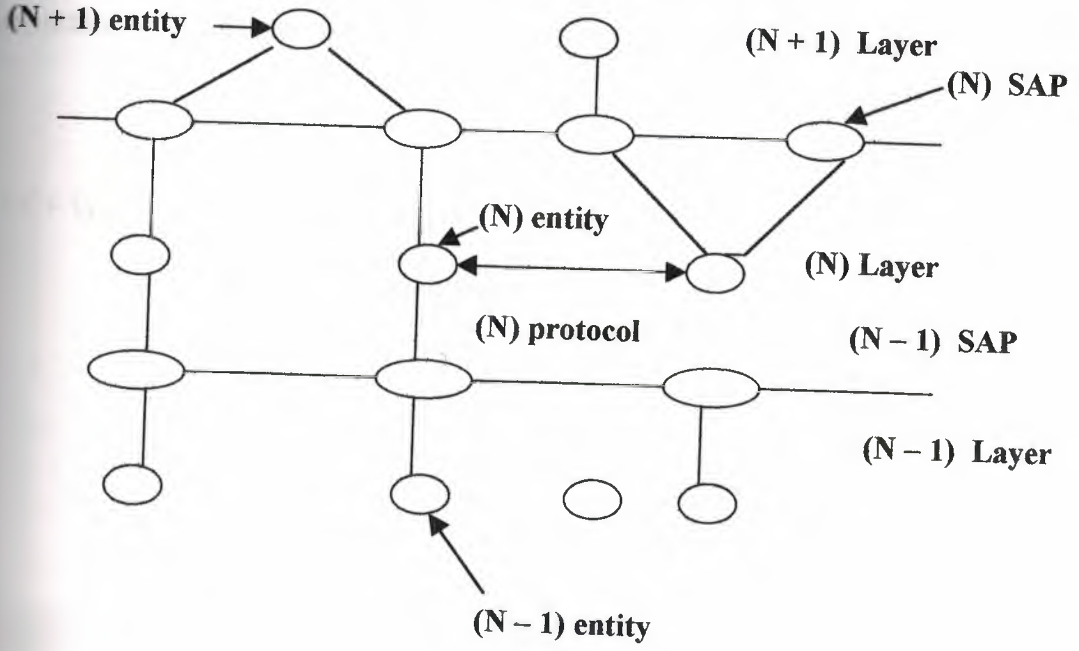


Figure 2.5 Layers, entities and SAPs.

- At most one entity is responsible for supporting a SAP;
- No more than one (N + 1) entity may access services through an (N) SAP at a time;
- An entity may support any number of SAPs; and
- An (N + 1) entity may access services available at more than one (N) SAP.

Note that the association between the user (N + 1) entity and an (N) SAP is not necessarily permanent. It could dynamically change. The association between an entity and the SAPs at which it provides services is also not fixed.

The nature of services provided by a layer is specified in terms of the set of primitives (atomic actions) that an (N + 1) entity or an (N) entity may issue at an (N) SAP.

2.2.3 Protocols

A major concern of OSI is with communication between peer entities (of the same rank). For the case where peer entities reside within the same open system, there may exist a direct path or an interface between them, in which case such communication is considered to be outside the scope of OSI. In the absence of such an interface their communication is governed by procedures that are identical to those that are applicable to communication between peer entities residing in different open systems. Clearly, there exist no direct communication path between two peer entities when they reside in different open systems, except at the lowest layer, the transmission media. Thus, two (N + 1) entities wishing to communicate with each other must rely on communication services provided by the (N) layer. This they do by accessing (N) services at their respective (N) SAPs, the latter being supported by two corresponding (N) entities.

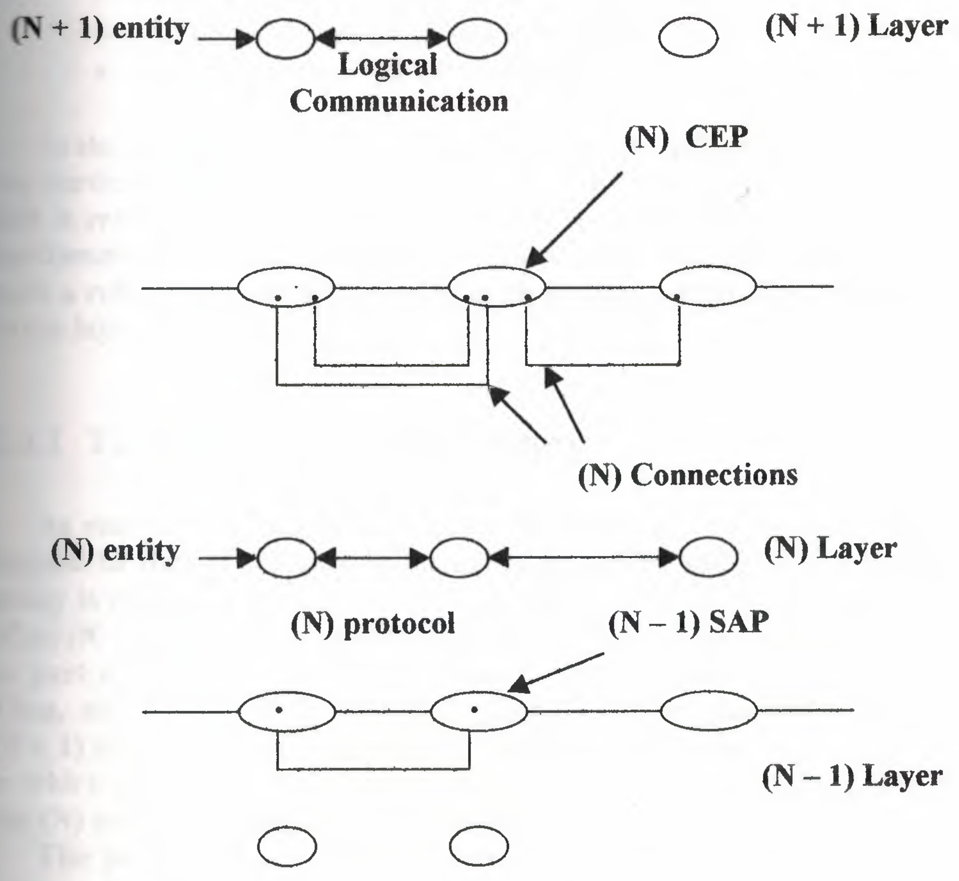


Figure 2.6 Connections and connection-endpoints.

As shown in Figure 2.6, such connections may be established between the same pair of SAPs, or even between different SAPs. All connections established via the same SAP are supported by the corresponding (N) entity. To enable the supporting (N) entity and the attached (N + 1) entity to distinguish between various connections established through the same (N) SAP the notion of connection-endpoints is introduced. For each connection to connection-endpoints are defined one for each end of the connection. Such an (N) connection-endpoint ((N) CEP) terminates an (N) connection at an (N) SAP. Thus, an (N) CEP associates three objects, namely, an (N + 1) entity an (N) entity and an (N) connection. A reference to a CEP by the supporting entity immediately identifies, for the (N +1) entity, the (N) connection and vice-versa.

2.3 Identifiers

A notation for uniquely identifying objects, including entities, SAPs, and connections, is considered in this section. We also discuss techniques for maintaining correspondences between entities and the SAPs to which they are attached, or between SAPs from adjacent layers.

To be able to uniquely reference an object anywhere within the network, the OSI architecture requires that each object within the OSI have a unique identifier, or a name. Identifiers associated with entities are called titles, while service-access points are identified using addresses. A connection is primarily identified by its endpoints using a connection-endpoint-identifier (CEP-identifier) for each CEP. To be sure:

- An (N) entity has an (N) title,
- An (N) SAP has an (N) address, and
- An (N) has an (N) CEP identifier.

In the above, the "(N)" suggests that such identifiers have a significant that is local to the particular layer, the (N) layer. Also, notice that an (N) entity has simply a title. This title is referred to as a global-title which is unique within the domain of the entire OSI environment. One may instead use a local-title to uniquely refer to an entity. But when such a reference is made the scope must be clear or obvious. Usually the scope is limited to the layer in question.

2.3.1 Titles, Addresses and Directory

As one consequence of the above, one does not refer to the address of an entity, but instead to that of the title of an entity or to the address of a SAP through which the entity is reachable or to which it is attached. The latter binding between the global-title of an (N + 1) entity and the (N) address is in a directory maintained by the (N + 1) layer as part of its (N + 1) functions. Such a directory referred to as an (N + 1) directory. Thus, an (N + 1) entity wishing to, for example, establish a connection with another (N + 1) entity may consult the (N + 1) directory to determine the (N) addresses of a SAP to which the remote (N + 1) entity is attached. It may then make this address available to the (N) entity that supports the local SAP.

The purpose of an (N) directory is to maintain a listing of the binding in existence between the titles of (N) entities and (N – 1) addresses of (N – 1) SAPs. Such a directory may be consulted by any (N) entity and is treated as a layer wide directory. In a dynamic environment, in which buildings change with time, the directory entries have to be updated. A number of implementation issues arise when we consider how directories may be implemented, managed updated and accesses in an OSI environment.

2.3.2 Address-Mapping

Next, we discuss the concept of an (N) address-mapping, and two different ways of implementing it. But, we first discuss an application where it is relevant for an (N) entity to identify the (N – 1) SAP that a remote entity uses the support an (N) SAP. Consider again the process of establishing connections between peer (N + 1) entities. For, this, an (N + 1) entity makes available an address to the entity that supports its local SAP. This supporting entity must now establish a connection, if necessary with the entity that supports the remote (N) SAP. It is truly not necessary for it the first determine its title and sub-sequentially the (N – 1) address of the (N – 1) SAP to which the remote (N) entity is attached. Only the latter would suffice. This done using the (N) address-mapping function.

The (N) address-mapping, a function implemented within an (N) layer, provides the mapping between an (N) address and the (N – 1) address associated with the (N) entity. There are two kinds of address-mapping functions that may be defined:

- Hierarchical (N) address-mapping; and
- (N) address-mapping by table.

Hierarchical address-mapping is somewhat simpler to implement, but may only be used in a layer for which every address is mapped onto one (N – 1) address, and where such associations are permanent. In hierarchical mapping, a number of addresses are mapped onto a single (N – 1) address (see Figure 2.7). These restrictions then enable a simple mapping function. An (N) address is composed of two parts:

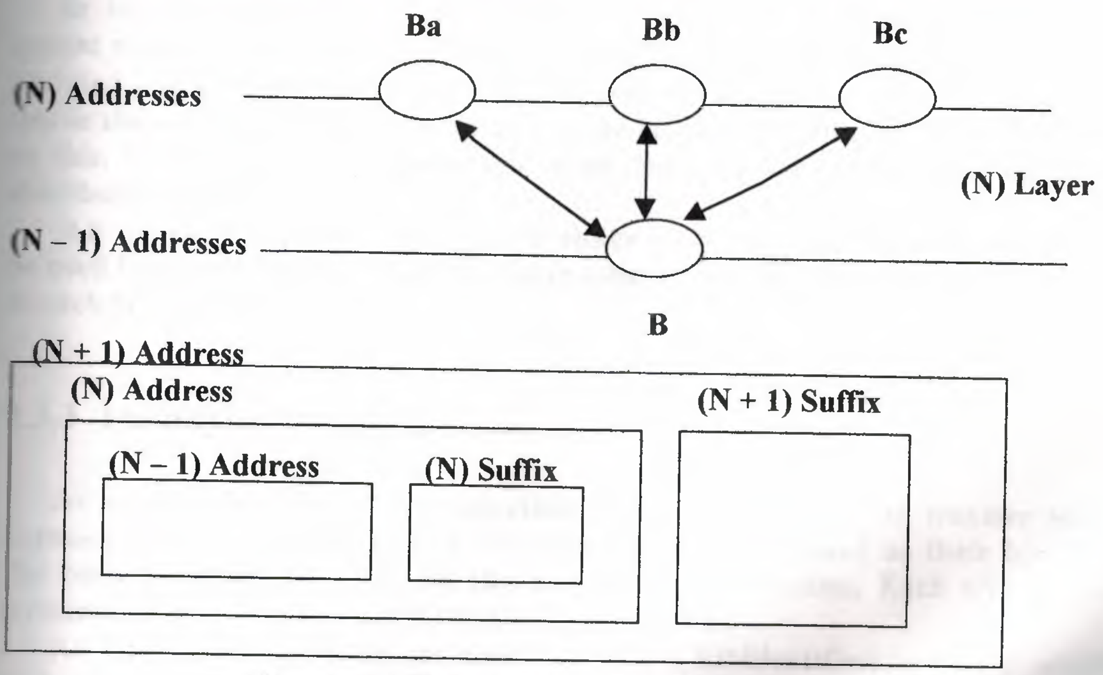


Figure 2.7 Hierarchical (N) address-mapping.

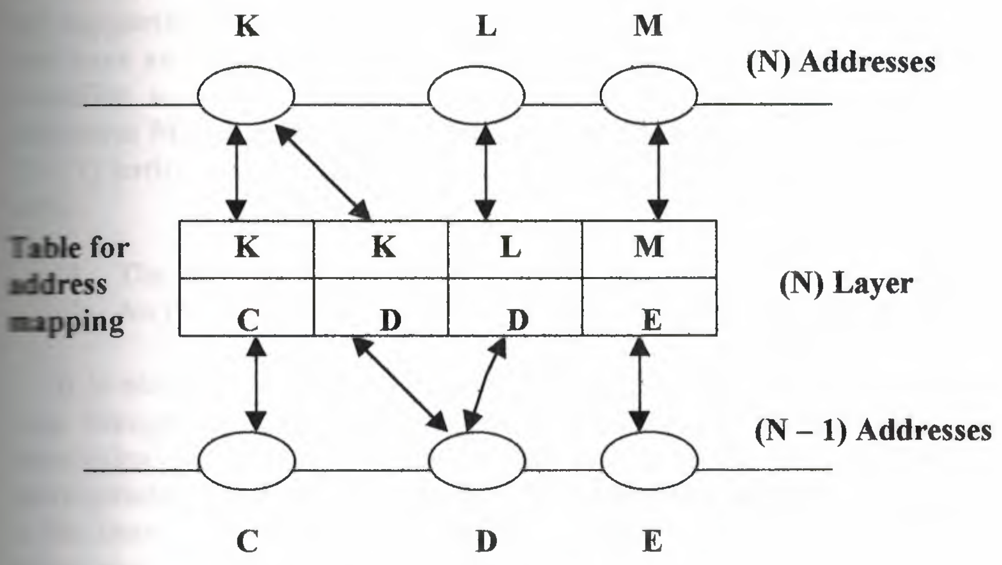


Figure 2.8 An example table for (N) address-mapping.

- An (N – 1) address of the (N – 1) SAP which supports the (N) entity, which in turn supports the (N) SAP, and
- An (N) suffix which uniquely identifies the particular (N) SAP with in the domain of all SAPs supported by the (N) entity.

As such, the (N – 1) address of the supporting (N – 1) SAP may be obtained by simply stripping of the suffix from the (N) addresses. A table lists, for each (N) address, the collection of all (N – 1) addresses of which it maps. An example of a table of address-mapping is given in Figure 2.8. Address–mapping by table permits greater flexibility, although its implementation would, in general, be more complex. None of two restrictions mentioned in the context of hierarchical mapping are applicable.

In implementing a table-based address-mapping, the mapping within each open system may have to be defined using a local table. Such a table may be stored, managed, updated and accessed within the open system. A collection of all such tables in a layer define the complete address-mapping for the layer. A distributed implementation, such as this, raises many implementation issues that are similar to those encountered in distributed databases.

It is worth noting that within a layer either of the two address-mapping schemes may be used irrespective of the scheme used in other layers. In this regards, address-mapping in each layer is independent.

2.3.3 Identifying Connections

As noted in Section 2.2.4, connections are a common way to transfer information between peer (N + 1) entities. An (N) connection is established on their behalf between the corresponding (N) SAPs by the supporting (N) entities. Each (N) connection is terminated at each end in an (N) connection-endpoint.

An (N) connection-endpoint-identifier ((N) CEP-identifier) uniquely identifies an endpoint of an (N) connection. It allows the (N + 1) entity, attached to the (N) SAP, and the supporting (N) entity to distinguish a connection from other connections that may also have an endpoint within the same SAP. Thus, it is sufficient to ensure that a CEP identifier is unique within the domain of the particular SAP. However, the OSI Reference Model insists that a CEP identifier be unique within the scope of the attached (N + 1) entity, instead. It, therefore, views the (N) CEP identifier to be consisting of two parts:

1. The address of concerned (N) SAP, and
2. An (N) CEP suffix, which is unique within the scope of the SAP.

It is obvious that the CEP identifiers at the two ends of a connection are distinct, even though the CEP suffix at the two CEPs may be the same. Furthermore, the association between a CEP identifier and the CEP is meaningful as long as the corresponding connection exists. The CEP identifier is assigned by the supporting entity at the time of connection establishment, and loses significance with the release of the connection.

In the past, we have not referred to identifiers for connections themselves. Identification of connections is required so that the supporting entities may distinguish one connection from the others that they support. For such purposes each connection is said to have an (N) protocol-connection-identifier. This identifier must be unique within the scope of the pair of supporting (N) entities.

The OSI Reference Model recognises the need for yet another identifier for each connection. The scope of such an identifier is however, different from that discussed above. Each connection is additionally identified by an (N) service-connection-identifier. The latter serves to bind three objects, namely, the two corresponding user (N + 1) entities and the (N) connection. Thus, communicating (N + 1) entities are able to distinguish one connection from others, but scope of the identifier is limited to the two (N + 1) entities.

2.4 The Nature of Data Units

This section introduces a notation for different types of data units exchange between peer entities or between entities from adjacent layers. The discussion brings out the distinction between information exchanged only for the purposes of co-ordination and user-data, the latter being the focus all communication.

Exchange of information may take place either between two peer entities or between an (N + 1) entity and an (N) entity that are attached to the same SAP. The nature of information exchanged between a pair of entities may be classified into two types:

- User-data, and
- Control information.

Transfer of data is the prime objective of all communication between entities. But, entities also need to exchange control information which enables them to co-ordinate their operations so as to exchange data. Examples of control information include address of destination, sequence number associated with data being exchanged, acknowledgement information. More generally, control information provides a description of the state of the entity participating in information exchange, or additionally describes user-data being exchanged.

2.4.1 Data Units

Recall that information exchanged between an (N + 1) entity and an (N) entity takes place across an interface, while information exchanged between two peer entities is governed by an (N) protocol. In view of the distinction between control information and data, it is pertinent to define four different types of data units:

- Protocol-control-information: information exchanged between peer entities to co-ordinate their joint operation;
- User-data: data transferred between (N + 1) entities for whom the (N) entities provide services;
- Interface-control-information: information transferred between an (N + 1) entity and an entity to co-ordinate their joint operation;
- Interface-data: data transferred from an (N + 1) entity to supporting (N) entity for transmission to a corresponding (N + 1) entity, or vice-versa.

It is often the case that data is transferred along with control information. We, therefore, require two additional definitions:

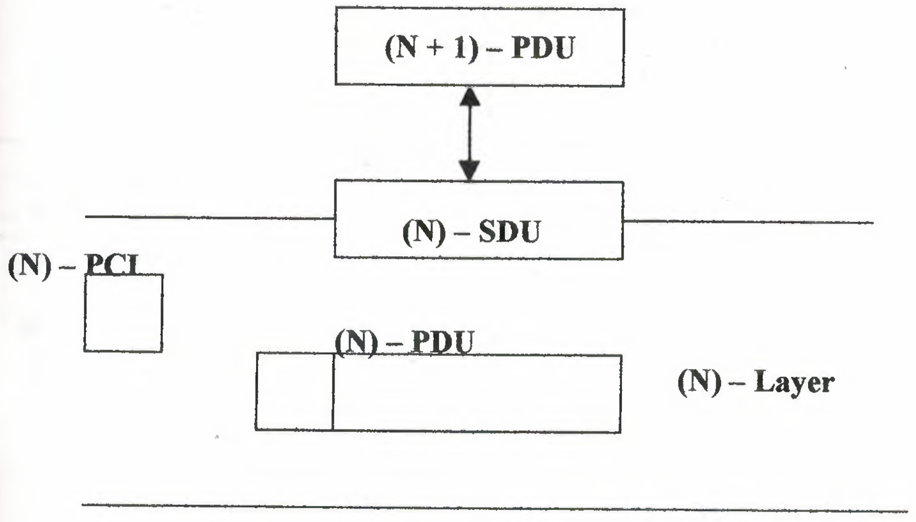
- (N) protocol-data-unit ((N)-PDU): information exchanged between peer entities, which consist of control information as well as user data;
- (N)-interface-data-unit: information exchanged between an (N + 1) entity and an (N) entity across a SAP, which consists of control information as well as user-data.

An (N) protocol (governing communication between peer entities) specifies the set of PDUs. It is from this set that an entity selects a relevant PDU to transfer control information and possibly data. On the other hand, a description of services does not include specification of the set of interface-data-units. Instead, it is recognised that exchange of information between an (N + 1) entity and an (N) entity is across an interface within an open system. It is, therefore, not subject to standardisation within OSI. The definition of these information types is included within the OSI Reference Model to distinguish these from (N) service-data-units.

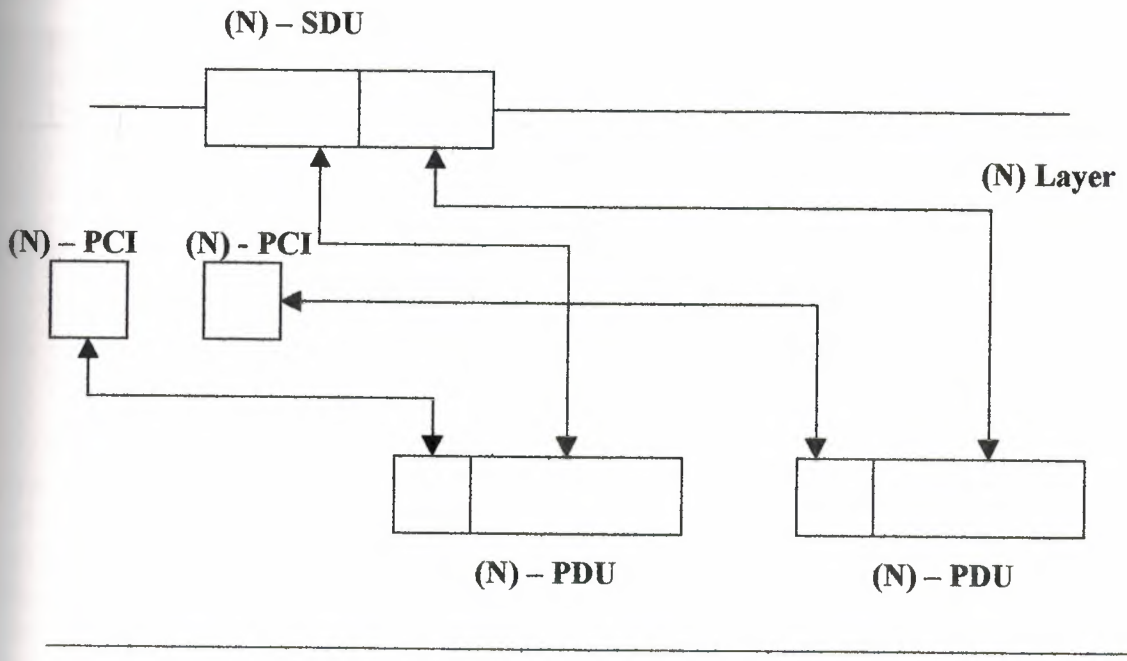
An (N) service-data-unit ((N)-SDU) is interface-data whose identity is preserved from one and of a connection to the other. It is immaterial how an (N)-SDU is exchanged between a pair of (N + 1) entity and (N) entity, as long as boundaries between SDUs are preserved. In fact, an SDU may well be exchanged in one or more interface-data, or a number of SDUs may be exchanged within an interface-data. (Also note that SDUs only contain data.)

There may be occasions when an (N + 1) entity may wish to communicate a small amount of data on a priority basis. This need is well recognised within OSI. As such a layer may provide expedited data transfer service. Such a service accepts an (N) expedited-service-data-unit ((N)-expedited-SDU) and transfers it over a connection possibly on a priority basis. The (N) layer may not be in a position to guarantee its delivery within a pre-specified time delay. It does, however, ensure that an (N)-expedited-SDU will not be delivered after any subsequent SDU or expedited-SDU.

Recall that a layer provides a service that enables communicating (N + 1)entities to exchange data, in fact, (N + 1)-PDUs. An (N + 1)entity hands over an (N + 1)-PDU to the supporting (N) entity at the (N) SAP in the form of an (N)-SDU. This SDU is delivered to remote (N + 1) entity at a corresponding SAP. The manner in which such an SDU gets entities. The sending (N) entity treats this as user-data and forms an (N)-PDU by appending to it the relevant protocol-control-information as dictated by the protocol. This mapping of (N + 1)-PDU onto an SDU and of an SDU onto a PDU is illustrated in Figure 2.9(a). Therein, we have assumed that neither segmentation, blocking, nor concatenation is performed. Other forms of mapping are discussing in the remaining part of this section.



(a): Neither segmentation, blocking nor concatenation.

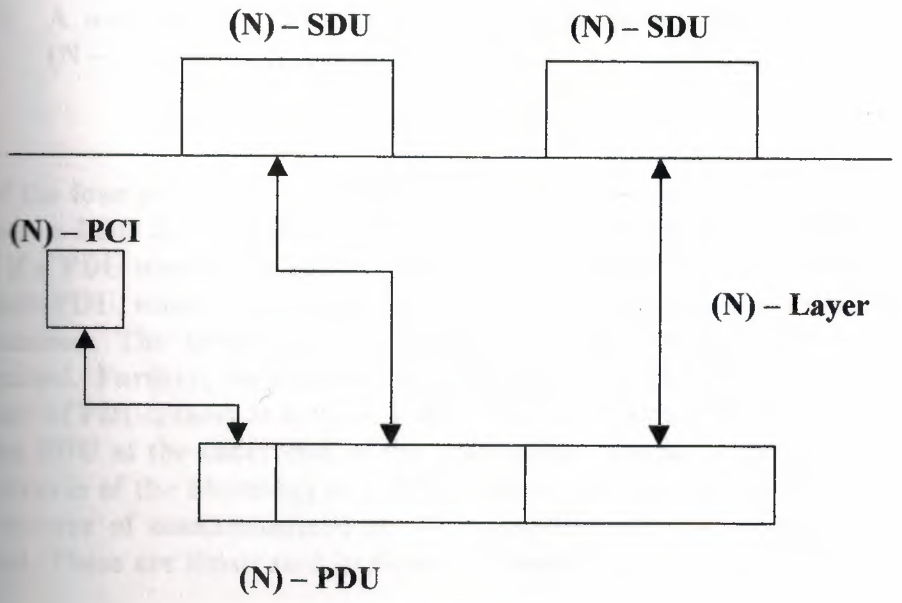


(b): Segmentation and reassemble.

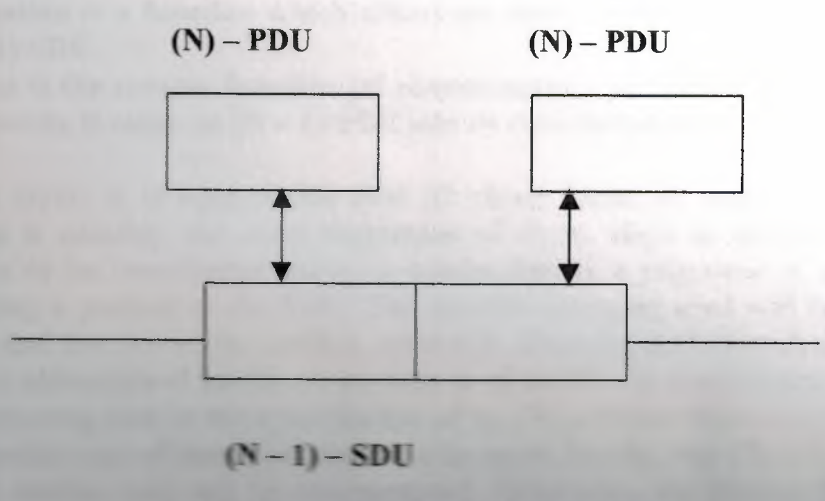
Figure 2.9 Mapping between different data units in adjacent layers.

2.4.2 Segmentation, Blocking and Concatenation

The OSI Reference Model does not place any constrains on the size of data units. This is primarily to allow implementations to define their own constraints on permissible size, a decision that may be based on available buffer size. To support varying length of data units, the OSI Reference Model permits a data unit to be mapped onto a number of data units, or for a number of data units to be mapped onto one data unit. Thus, one may consider the following possibilities:



(c): Blocking and de-blocking.



(d): Concatenation and separation.

- **An (N) – SDU is segmented and subsequently mapped onto a number of (N) – PDUs;**

- **A number of (N) – SDUs are blocked together and mapped onto a single (N) – PDUs;**

- **An (N) – PDU is broken down into a number of sub-PDUs, each of which is mapped onto a different (N – 1)-SDU;**

- **A number of (N)-PDUs are concatenated together and mapped onto a single (N – 1)-SDU.**

Of the four possibilities listed above, the third is recognised to be meaningless. This is so since a PDU is composed of two parts, protocol-control-information and user-data. Now, if a PDU were to be broken down into a number of sub-PDUs, then, except for the first sub-PDU, none of the other sub-PDUs would have any associated protocol-control-information. The other three forms of mapping between PDUs and SDUs are well recognised. Further, corresponding to segmenting of SDUs, or mapping them onto a number of PDUs, there is a reverse mapping, or re assembly, of the corresponding PDUs into an SDU at the other end of the connection. Similarly, mechanisms for de blocking (the reverse of the blocking) of a PDU into the corresponding SDUs, and for separating (the reverse of concatenation) an (N – 1)-SDU into corresponding PDUs need to be defined. These are illustrated in Figure 2.9 and formally defined below.

1. **Segmentation is a function performed by an (N) entity by which it maps one (N)-SDU into multiple (N)-PDUs.**
2. **Re assembly is the reverse function (of segmentation) whereby a corresponding (N) entity maps corresponding multiple (N)-PDUs into one (N)-SDU.**
3. **Blocking is a function performed by an entity by which it maps multiple (N)-SDUs into (N)-PDU.**
4. **De blocking is the reverse function (of blocking) whereby the corresponding multiple (N)-SDUs.**
5. **Concatenation is a function which allows an entity to map multiple (N)-PDUs into one (N – 1)-SDU.**
6. **Separation is the reverse function (of concatenation) performed by a corresponding entity whereby it maps an (N – 1)-SDU into its corresponding multiple (N)-PDUs.**

Within a layer, it is conceivable that all three forms of mapping may be used. Segmentation is possibly the most important of these, since it allows an SDU of an arbitrary size to be transferred across a connection as a sequence of multiple PDUs, each containing a portion of the SDU. The specific mapping used will be a function of the protocol and the size of the buffers available. Blocking and concatenation permit a more efficient utilisation of an (N) connection or of an (N – 1) connection, respectively. It is worth mentioning that in the specification of an (N) protocol there may be constraints placed on whether any of these functions can be used. Surely, two (N)-SDUs destined for different (N) entities may not be concatenated. Otherwise, the reverse functions of de blocking or separation cannot be carried out ! It is, therefore, relevant to constrain the

use of these functions to map data units that pertain to communication between the same pair of (N) entities.

2.5 Connection-Based Data Transfer

In this section we discuss the more common approach to transfer data over an established connection. Here, we describe in some detail the procedures to establish or to release connections, aside from functions relating to data transfer that are generally preferred to be implemented.

As mentioned earlier in Section 2.2.4, two (N + 1) entities may communicate with each other over an (N) connection established and maintained on their behalf by supporting (N) entities between the corresponding (N)-SAPs. Such a connection is, in fact, an association (however temporary) between three parties, namely, the two (N + 1) entities and the (N)-Layer. The establishment of this association enables the two (N + 1) entities to, firstly, express agreement (or disagreement) on their willingness to communicate with each other. Further, while agreeing to do so, they also decide upon the syntax and semantics (N + 1)-protocol of all information exchanges that would take place over the connection. The process of establishing a connection also enables the communicating (N + 1) entities to initialise themselves to a mutually known global state so that subsequent exchanges of information maybe inter-pretend and acted upon an accordance with the agreed (N + 1)-protocol.

Since the (N)-Layer is actively involved in establishing and maintaining the (N) connection, the agreement includes a commitment on the part of the layer to support the connection to the extent it is able to. This particularly so in respect of the nature of the connection and the quality of services provided. Towards the latter, the relevant entities determine for themselves as to how they can best support the connection by selecting an appropriate (N)-protocol. The supporting entities may themselves need to establish an (N – 1) connection over which all communication pertaining to the particular (N)-connection takes place. Assignment of such resources, including that of message buffers, would also be done at the time of establishment of the connection.

Connection-oriented interaction between (N + 1) entities proceeds through three distinct phases: connection establishment, data transfer, and connection release. Data transfer may only take place once a connection has been establishment. The connection is preferably released once data transfer is complete since committed resources can be re-allocated for use with other connections.

2.5.1 Connection Establishment

The manner in which connections are established or released varies from layer to layer. Similarly, procedures that govern data transfers are dependent upon the nature of service requested or offered over the particular connection and upon the selected protocol. However, there are certain aspects that are common to all layers. These are discussed below.

Before attempting to establish a connection, the (N + 1) entity initiating the connection must know the title of the (N + 1) entity it wishes to communicate with. With that title, its (N)-address may be obtained from the corresponding directory. The connection establishment request is then initiated using the address.

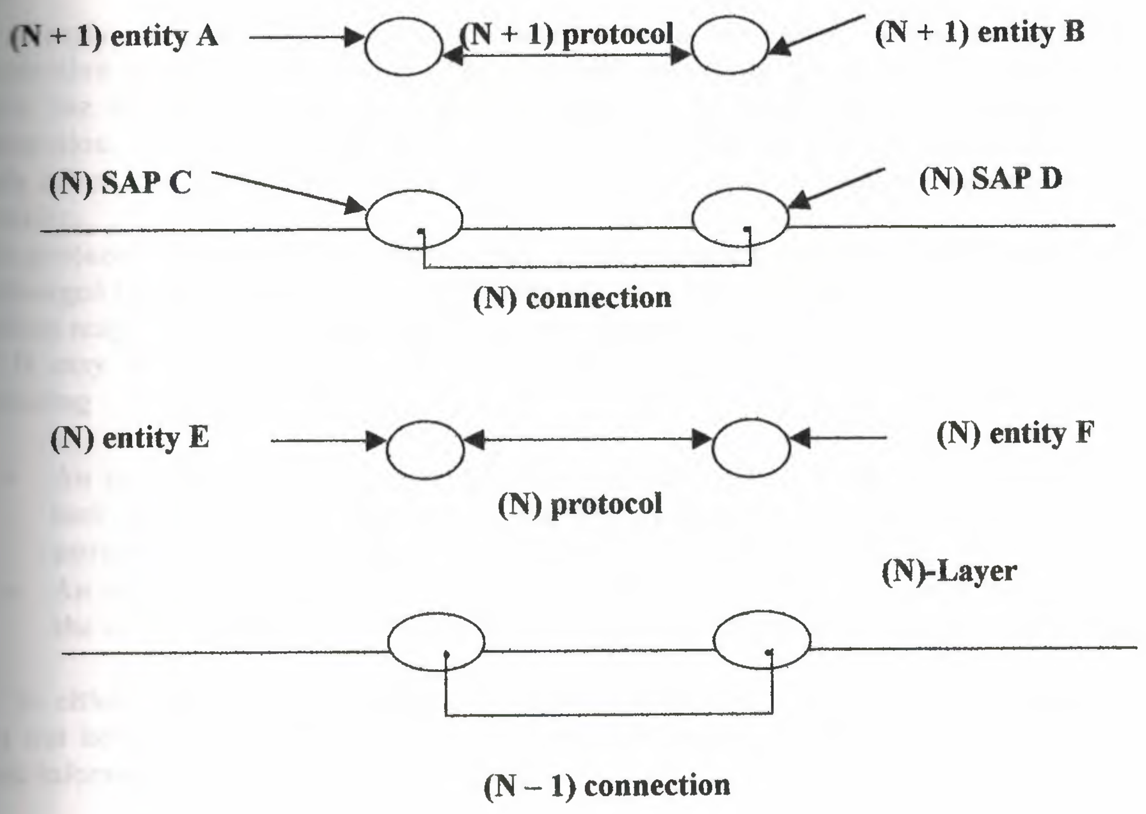


Figure 2.10 Establishment of an (N) connection.

The connection establishment procedure is illustrated in Figure 2.10, where an (N + 1) entity A initiates the establishment of connection with an (N + 1) entity B. The connection is established between the corresponding SAPs C and D. It is through the attached (N) entities E and F that the (N)-Layer provides connection-oriented services at the two SAPs. The establishment procedure, typically, involves the following six steps:

1. The (N + 1) entity A, while initiating the establishment of an (N)-connection, specifies, together with the request at its (N)-SAP C, the (N)-address of the (N)-SAP D to which the responding (N + 1) entity B is attached.
2. The supporting (N) entity E communicates the request to the (N) entity F at the other end.
3. The (N) entity F informs the responding (N + 1) entity B (at the (N)-SAP D) of the incoming request for connection establishment with the (N)-address of the SAP-C to which the initiating (N + 1) entity A is attached.
4. If the establishment of the (N) connection is acceptable to the responding (N + 1) entity B, it simply informs its supporting (N) entity F at its (N)-SAP D.
5. The (N) entity F communicates this acceptance by the (N + 1) entity B to the (N) entity E at the initiator's end.
6. The (N) entity E conveys to the initiating (N + 1) entity A the acceptance obtained from the responding (N + 1) entity B.

Clearly, the two (N + 1) entities interact with the layer during the process of connection establishment. As such they may also negotiate between themselves and the layer the optional services (and their quality) to be provided over the established connection. Furthermore, since the supporting entities themselves communicate with each other, they may select the appropriate protocol to be used for subsequent data transfer.
The protocol between the two entities may permit a limited amount of user-data to be exchanged as part of connection establishment. As a consequence, the two (N + 1) entities may fix the (N + 1) protocol to be associated with subsequent data transfers.

It may be noted that the attempt to establish a connection may fail any reason, including

- An unwillingness on the part of the responding (N + 1) entity, either because of lack available resources, or its inability to work with the type of connection purposed by the initiating (N + 1) entity or offered by the layer;
- An inability on the part of the layer to allocate required resources, or to provide the optional services (or their quality) requested by the initiating (N + 1) entity.

In either case, the connection establishment procedure is terminated prematurely, but not before all parties involved in the establishment process up to that stage have been informed of the failure of the attempt.

2.5.2 Multiplexing and Splitting

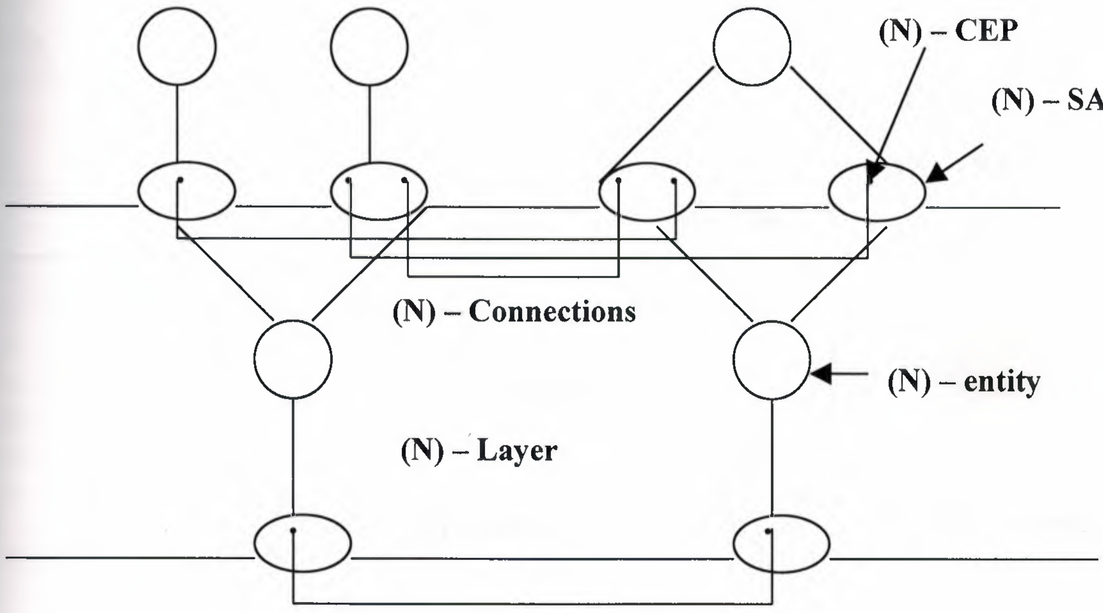
A major requirement of a layer that provides connections is that supporting entities should be able to communicate each other. Either they are within the same open system and a direct (outside the OSI environment) interface exists between them, or they communicate over an (N – 1)-connection. Such an (N – 1)-connection, if does not already exist, will need to be established before any connection-related communication between the entities may take place. But, in case the protocol of (N – 1)-layer permits, (N – 1)-user-data may be exchanged during its establishment. As a consequence, an (N) connection could be established simultaneously with that of (N – 1) connection. Of course, before two (N – 1) entities communicate there must exist an (N – 2) connection, and so on. A physical transmission media must be available at the bottom-most layer.

Another issue related to the above is that of mapping connections onto (N – 1) connections. It is recognised, within OSI, that PDUs relating a number of (N) connections may be transmitted on the same (N – 1) connection, as long as the (N) connections are supported by the same pair of entities. This is referred to as multiplexing of (N) connections is done at one end, then surely the reverse operation of de-multiplexing must be performed at the other end. Multiplexing may be absolutely essential those cases where only one (N – 1) connection can be established. Further, multiplexing enables a more efficient and often more economical use of an (N – 1) connections with the particular (N) connection is called recombining. Figure 2.11 illustrates the concepts of splitting an recombining.

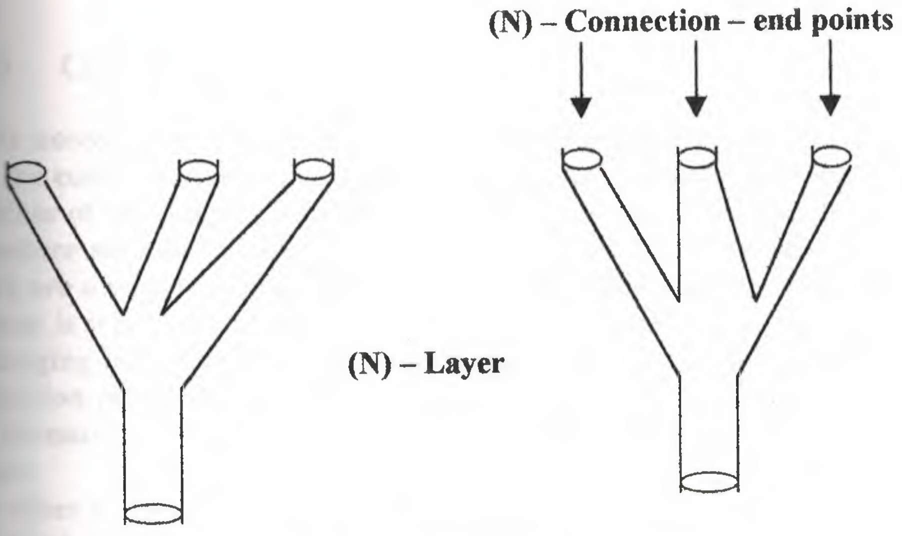
The use of multiplexing or splitting calls for implementation of a number of sub-functions within the (N)-layer. Some of these are listed below.

1. A means to identify (N)-PDUs that pertain different (N) connections, but which are sent as (N – 1) user-data over the same (N – 1) connection. This identification is done by associating with each PDU a protocol-connection-identifier.
2. Mechanism to schedule the transmission of (N – 1) user-data from different (N) connections over the same (N – 1) connection. Such a mechanism would also incorporate the means to control the rate of flow of user-data originating from different (N) connections.
3. A means to schedule the transmission of (N – 1) user-data from an (N) connection over different (N – 1) connections.
4. A mechanism to re-sequence (N)-PDUs, associated with an (N) connection, in case they arrive out of sequence. The latter may be the case when they are transmitted over different (N – 1) connections, and even though each (N – 1) connection may guarantee in-sequence delivery of (N – 1) user-data.

The first two functions are needed only if multiplexing is supported, while the latter are required only to support splitting.

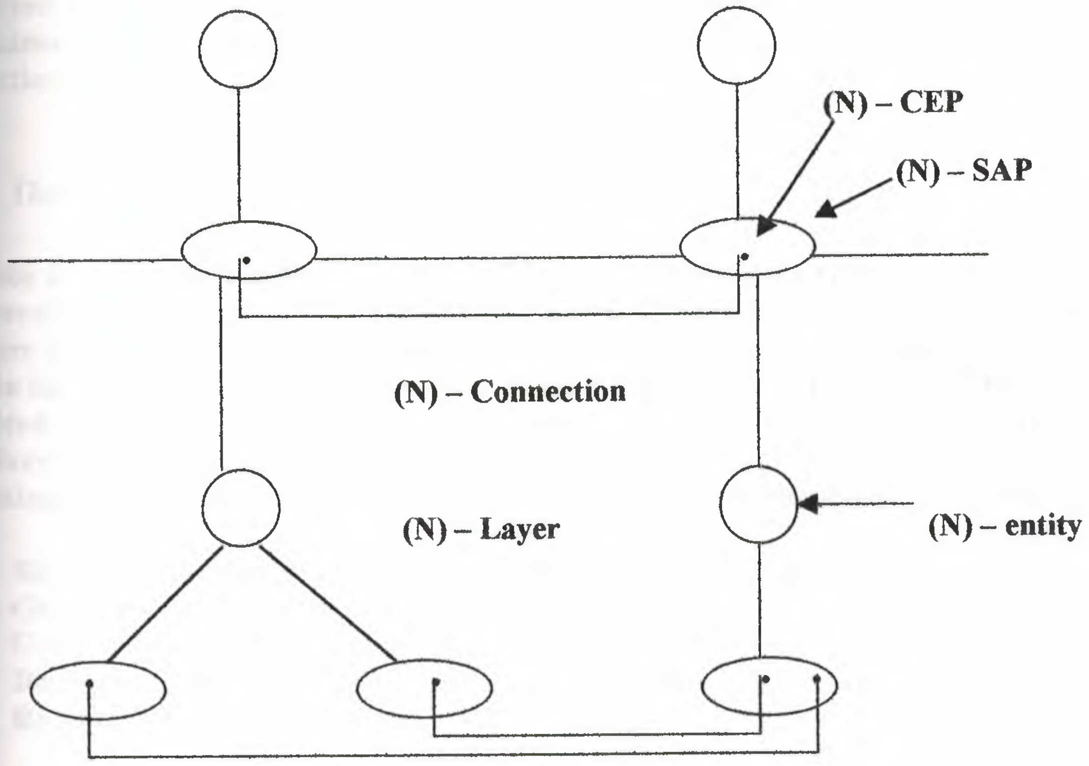


(a) The logical view.



(b) Connections viewed as pipes.

Figure 2.11 The concept of multiplexing and de-multiplexing.



(a) The logical view.

2.5.3 Connection Release

As noted earlier, a connection establishment attempt may be unsuccessful. In that case the connection is automatically released. Additionally, a connection may be released by either of the communicating (N + 1) entities, or by the supporting (N)-layer. A release procedure may be invoked by either party once the connection has been established. There are a variety of ways in which the connection may be released. The most graceful of these is where the communicating (N + 1) entities agree to release the connection by exchanging information in a manner very similar to the one described in the context of connection establishment. As part of that information exchange, the supporting entities also become aware of the connection release. Such a release procedure is termed orderly release.

The other variations of the release procedure are more abrupt and somewhat unilateral. As a consequence, there may loss of user-data. Either of the (N + 1) entities may decide to release the connection. Of course, the other parties, namely, the corresponding (N + 1) entity and the two supporting entities, do participate in the process, but have very little say in it. Similarly, either of the supporting entities may terminate the connection. The latter situation may arise when, for example, an entity detects a breakdown of the supporting (N – 1) connection breaks down. It is quite possible that the (N) connection is maintained while an attempt is made to re-establish an (N – 1) connection.

It is not necessary that the supporting (N – 1) connection be released once the supported (N) connection is released. The (N – 1) connection may continue to be maintained to support other connections that currently use it, or to support future connections.

2.5.4 Data Transfer

Once a connection has been established, user-data originating at an (N + 1) entity is made available to the supporting (N) entity in the form a sequence of SDUs. These data units are then transferred to its corresponding peer entity which subsequently delivers them to the corresponding (N + 1) entity again in the form of a sequence of SDUs and expedited-SDUs. The only constraint placed thus far is that an expedited-SDU may not be delivered after any subsequent SDU or expedited-SDU. A number of issues pertaining to the transfer of such a sequence still remain to be discussed. This include:

- Regulating the rate of flow of user-data over a connection.
- Guaranteed delivery of SDUs in the proper sequence.
- Confirming the delivery of user-data to the destined (N + 1) entity.
- Detection of errors and loss of SDUs; and recovery.
- Re-initialising the connection.

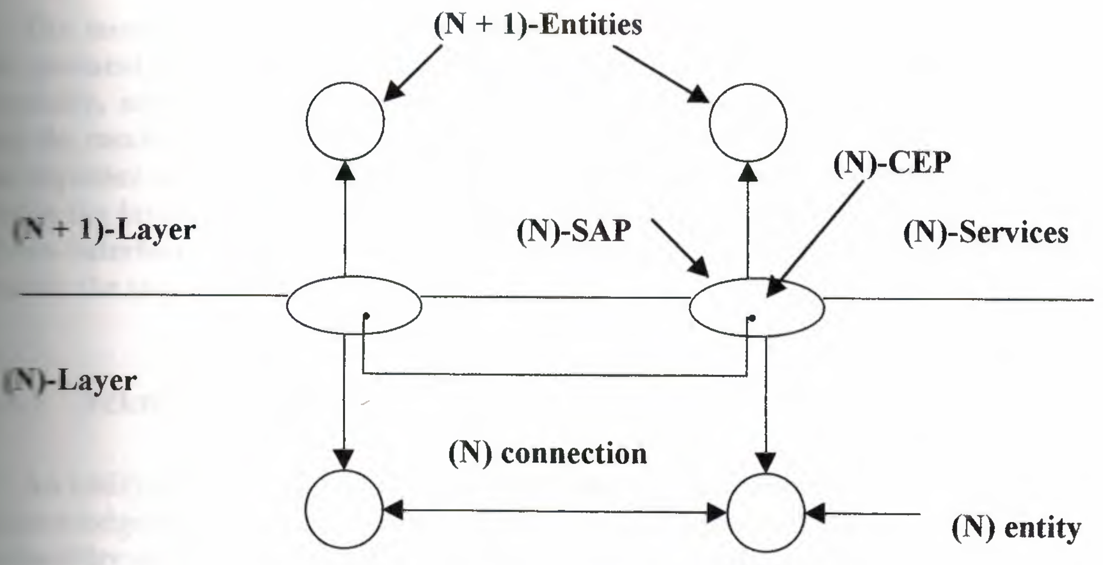


Figure 2.13 An (N)-connection viewed as a path consisting of three elements.

2.5.5 Flow Control

Our first concern is with limiting the rate at which user-data is made available to that which can conveniently be supported over a connection. However, communicating (N + 1) entities generate data at a rate dictated by application, as well as the (N + 1) protocol they use. One scheme is to explicitly limit the rate at which user-data at a rate dictated by the application, as well as the (N + 1)-protocol they use. One scheme, however, does not dynamically adjust to changing conditions in terms of availability of communication and computing resources within the layer and those below. As such, a scheme, referred to as flow control, is used *which dynamically limits the amount of data* that is made available or transfer over connection may be viewed as a path consisting of three segments, as indicated in Figure 2.13.

Flow control between peer entities limits the rate at which user-data (within (N)-PDUs) is exchanged between them. This peer flow control is defined as part of the protocol. The protocol may also limit the amount of user-data that may be contained in a PDU. Similarly, at each SAP, there may exist some form of flow control on user-data exchanged between a supporting entity and the attached (N + 1) entity. The specification of the nature of such interface flow control is considered to be outside the scope of OSI, and as such implementation dependent.

Expedited-SDUs are not subject to the same flow control as are SDUs. Where necessary, a separate flow control would be applied to the transfer of expedited-SDUs.

2.5.6 Sequencing

Delivery of (N)-SDUs in the proper sequence is an important function of a layer. In its absence, the sequence of user-data delivered to the receiving (N + 1) entity may be different from the sequence of user-data obtained from the sending (N + 1) entity. This may happen for a number of reasons, including loss of user-data followed by the retransmission, or user-data moving along different physical (or even logical) paths.

The mechanism to achieve in-sequence delivery of user-data is specified as part of the protocol, whenever the corresponding function is required to be implemented. Typically, user-data contained within each PDU is uniquely by the sending entity, so that the receiving entity can re-order the received PDUs, as necessary. This ensures that the sequence of user-data within PDUs is preserved across the segment of the connection within the layer. But that is not adequate. The sequence must also be preserved at each of two interfaces. The latter aspect is considered to be implementation dependent and outside the scope of OSI.

2.5.7 Acknowledgement

An entity sending information may, in certain applications, wish to receive an acknowledgement from the receiver. This may be necessary if there is a finite probability of information being lost or unduly delayed during transfer. In the context of a connection, the source and destination entities are, truly, the (N + 1) entities for whom the supporting entities establish and maintain the connection. OSI, however, is primarily concerned with acknowledgements to user-data (within PDUs) exchanged between the supporting entities. The mechanism to transfer acknowledgement information is specified, again, as part of the protocol. Such a specification normally requires identification of each PDU (only the ones that contain user-data need be identified).

The OSI does, however, recognise the need for a user (N + 1) entity to exchange acknowledgement information with its peer entity. This may be covered by the (N + 1)-protocol that operates between the peer entities. However, an additional mechanism is sometimes used to convey acknowledgement information between peer (N + 1) entities when they use a connection to transfer data. A receiver (N + 1) entity may request the supporting entity, at its end, that an indication suggesting confirmation of receipt be given at the other end to the sender (N + 1) entity. Such a mechanism is specified as part of the services that a layer may offer.

2.5.8 Error Detection and Recovery

Issues relating to preserving the sequence of SDUs across a connection and of acknowledgements are part of the larger issue of reliable data transfer. Reliability of data transfer refers to the requirement that SDUs be communicated without any error, loss of data, or duplication, and (possibly) in the same sequence with an acceptably high probability. Such reliable transfer must take place against all odds, including noise over transmission media, lack of computing resources, limited bandwidth, or excessive delays. Breakdown of transmission media, hardware faults, faults in software design, or non-conformity to OSI standards are examples of more serious failures. The latter may prevent communication of data altogether.

The OSI architecture and its protocols are concerned not so much with these impairments, but with detecting the occurrence of errors and of failures. Generally, if a layer detects an error, it makes every effort to recover from it using, for example, error detecting or correcting codes and, possibly a positive acknowledgement with re-transmission scheme. Normally, such attempts succeed, but when errors persist with high frequency, a re-initialisation of the connection may be undertaken in the hope that recovery may still take place. This re-initialisation, called reset, enables the entities to move back a pre-defined global state. There is, however, a finite probability that some errors may go unnoticed, in that case data may be lost or duplicated. If, in spite of all efforts, the layer is unable to recover from errors it simply signals a failure connection to the user (N + 1) entities. It is then the responsibility of the user (N + 1) entities to attempt a recovery or to abandon communication altogether.

Procedures to detect errors or failures and to recover from them are specified as part of the protocol. The procedures to reset a connection are also specified. A method by which a layer signals failure to user entities is specified as part of services. Typically, the layer must also provide a reason for the failure, if it is known, and whether such a condition is temporary or permanent.

2.6 Connection-Less Data Transfer

In this section we discuss an alternative approach to data transfer without first establishing a connection. It is emphasised that connection. It is emphasised that connection-less data transfer protocols are relatively simple since each data unit is self contained and totally unrelated to other data units.

In the previous section we have seen how connection-oriented data transfers between two user (N + 1) entities requires the establishment of an (N)-connection before user-data may be exchanged, and that this is to be followed by a connection release. Thus, connection-oriented data transfer may be characterised as follows:

1. Each connection has a clearly distinguishable lifetime as determined by the three distinct phases of establishment, data transfer, and release.
2. The successful establishment of an (N)-connection also establishes a three-party agreement between the two user (N + 1) entities and the layer which provides the connection-oriented service. This agreement indicates their mutual willingness to exchange data.
3. As part of connection establishment procedure, the three parties also negotiate use of certain optional services and parameter values to be associated with the connection. This enables each party to allocate resources that are required by particular connection.
4. (N)-SAP addresses are exchanged between user (N + 1) entities and the supporting (N) entities only during connection establishment. Subsequently, requests to transfer data over an (N)-connection (or to release it) make no reference to these addresses, but to the (N)-connection-endpoint-identifiers, one for each end.
5. (N)-service-data-units (as also (N)-expedited-SDUs) transferred over an (N)-connection are related to each other by virtue of their being transferred over the same connection. As such, it is relevant to discuss flow-controlled, or reliable transfer of sequence of (N)-SDUs.

Connection-less data transfer, on the other hand, is the transmission of independent, unrelated (N)-SDUs from one (N)-SAP to another in the absence of a connection. To support such data transfer an (N)-layer may offer connection-less (N)-service. Figure 2.14 illustrates how an (N + 1) entity A may transfer data to another (N + 1) entity B. The transfer is, typically, carried out in three steps:

1. The (N + 1) entity A passes the (N)-SDU across the local (N)-SAP C, to the supporting (N) entity E, together with the (N)-address of the (N)-SAP to which the destination (N + 1) entity B is attached.
2. The supporting (N) entity E transfers the (N)-user-data to the corresponding an (N) entity F which supports the (N)-SAP D, together with the addresses of the source of destination (N)-SAPs.
3. The (N) entity F passes the (N)-SDU across the (N)-SAP D to the attached (N + 1) entity B, together with the address of the (N)-SAP C to which the sending (N + 1) entity A is attached.

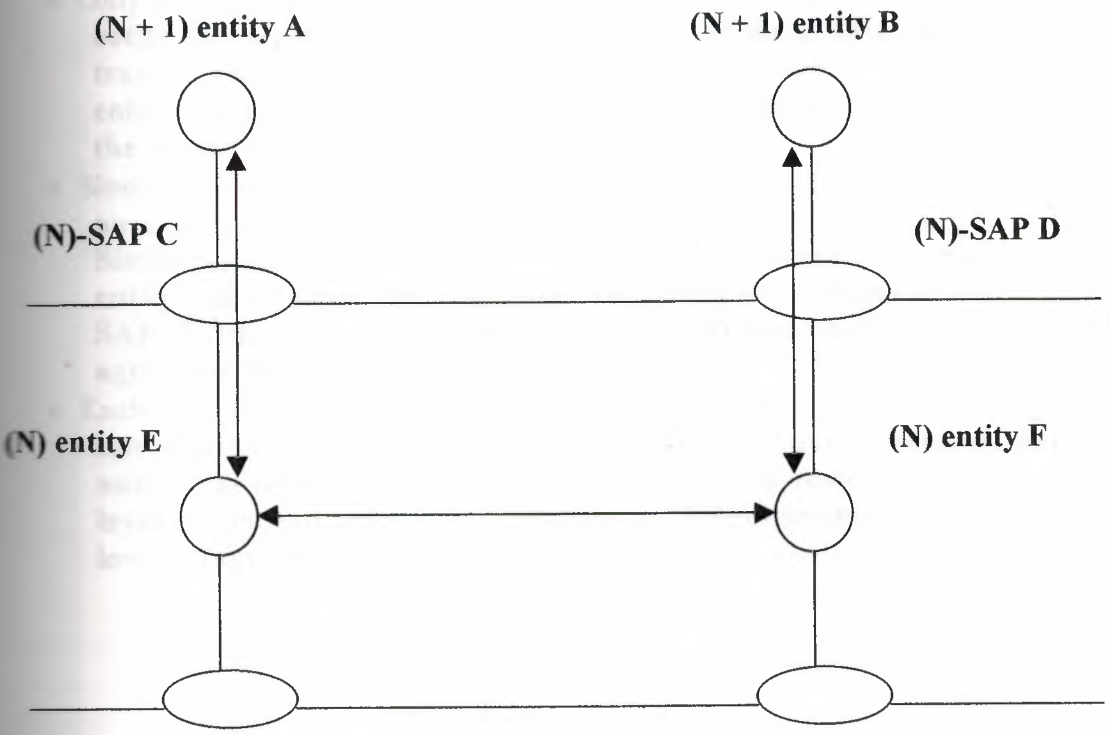


Figure 2.14 Connection-less data transfer.

The three step procedure ends with the delivery of the SDU to the destination (N + 1) entity. It is up to receiving (N + 1) entity to act upon the received data or to simply ignore it, depending upon a number of considerations. These may include the identify of the source (N + 1) entity, the nature of the data communicated, and its ability to interpret or process the received data. It is however, expected that each communicating (N + 1) entity has some prior knowledge of each other, particularly regarding their ability to interpret (syntactically as well as semantically) the data received. Any response generated subsequently by the receiving (N + 1) entity is similarly transferred, but as far as (N)-service is concerned, without any reference to a previous data-unit.

With the request to transfer data-unit, an (N + 1) entity may specify parameter values and options, such as transfer delay or acceptable rate of error, that are to be associated with the transfer of the particular SDU. Depending upon the manner in which the service implemented, the supporting entity may or may not be in a position to determine whether such a request cannot be met, then it may inform the requesting (N + 1) entity; otherwise it simply goes a head and makes a best effort to transfer the data. It may even be the case that data is not delivered to the destination (N + 1) entity, and neither the sending (N + 1) entity nor the supporting (N) entity becomes aware of this fact. The latter again depends upon how the two supporting (N) communicate between themselves. To be sure, communication between the supporting (N) entities may be connection-less or it may be over an established (N – 1) connection.

To summarise this discussion, connection-less data transfer exhibits the following characteristics:

- Only a single interaction between a user (N + 1) entity and the supporting (N) entity is required to initiate transmission of data. Once a request for data transfer has been made (or an (N)-SDU is delivered to the destination (N + 1) entity), no further interaction takes place between the user (N + 1) entity and the supporting (N) entity at its (N)-SAP.
- Since a connection is not established prior data transfer, data transfer is based on an a priori knowledge shared between the two communicating (N + 1) entities. Similarly, at each end, there is an a priori agreement between a user (N + 1) entity and its supporting (N) entity regarding (N)-services available at the (N)-SAP. Further, since negotiation is not performed, this a priori knowledge or agreement is not altered.
- Each data-unit is considered to be self-contained, in that the required address information is communicated together with the data. Independence of data-units from others implies that a sequence of data-units handed over to the (N)-layer at one end may not be delivered to the destination (N + 1) entity without loss or duplication or even in the same sequence.

CHAPTER 3

This chapter introduces the basic structure of the OSI architecture in terms of seven layers. The basic principles used in developing the layers are also introduced. Each layer in the OSI architecture is defined in terms of the services it offers and as a collection of required functions. The functions implemented within the layers enhance the services, in a step-by-step fashion, from those made available by the communication media to those required by user applications. This chapter also contains a brief discussion on OSI standards currently available, and their status regarding adoption by ISO and CCITT.

3.1 Introduction

The OSI architecture has been described in general terms in Chapter 2. This structure is centred around the concept of layers and has been used extensively in developing the OSI Reference Model. The model consists of the following seven layers (see Figure 3.1):

1. the Application layer,
2. the Presentation layer,
3. the Session layer,
4. the Transport layer,
5. the Network layer,
6. the Data Link layer,
7. the Physical layer.

The highest layer is the Application layer. It consists of Application entities that co-operate with each other to provide application-related services in an OSI environment. The lower layers, Physical through Presentation layers, provide services which make it possible for Application entities to communicate with each other. At the bottom, the Physical layer uses the communication media to exchange encoded bits of information.

The application entities are the final source and destination of all data. Some of open systems , however, simply perform the functions of relaying information from one open system to anther. Such a system, therefore, implements functions included in the three lower layers only.

In any case, as one goes up the layers one notice a layer-by-layer enhancement of services provided by each layer to entities in the next higher layer. This, as discussed in Chapter 2, is made possible by implementing in each layer a set of functions required the bridge the gap between the services that it provides and the services that it provides and the services available to it.

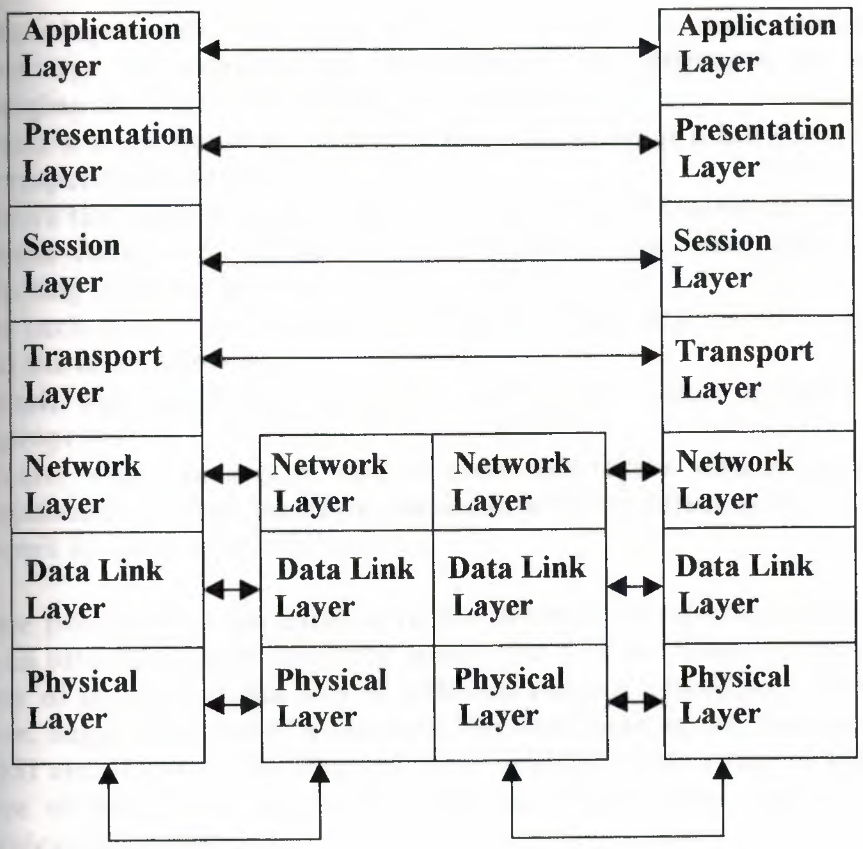


Figure 3.1 The seven layers of OSI.

As can be expected, there are variety of ways in which the OSI environment and its capability can be provided. Although the OSI Basic Reference Model prescribes the use of seven layers, the same capability can, in principle, be provided by fewer than seven layers, or using more than seven layers, or using more than seven layers. Further, the Reference Model defines, for each of the seven layers, the service that it provides to the next higher layer. In doing so, it implicitly specifies the collection of functions to be included in each layer. Here again, one may argue whether this is the most appropriate way of enhancing services from one layer to the next. This is equivalent to looking for alternative ways to partition the collection of functions necessary to provide OSI capabilities, and to assign them to different layers.

The above issues concerning the number of layers, and assignment of functions to each layer, have been defined and repeatedly used to obtain the seven-layer architecture, and to define the functionally of each layer. We shall briefly state these principles, and discuss how they relate to the design of the seven layers in the architecture. These are:

1. Have a reasonable number of layers to make the engineering task of system specification and integration no more difficult than necessary;
2. Define interfaces so that the description of services across the interface is simple;
3. Have a separate layer to handle functions which are clearly different in terms of the required processing or the supporting technology;
4. Include similar functions within the same layer,
5. Use successful experiences of the past in identifying the boundaries;

6. Create layers with well identified functions so that a layer can be modified to take advantage of technological developments in hardware or software, without changing the services of the adjacent layers;
7. Create a layer boundary where it may be useful at a later time to standardise its corresponding interface;
8. Ensure that each layer reflects a consistent level of abstraction in handling of data;
9. Permit changes to be made in the functions and protocols of a layer without affecting the other layers;
10. For each layer, have clear and well defined boundaries with only the layer above and the layer below it;
11. Permit the possibility of having sub-layers within a layer as necessary or appropriate;
12. Create, where necessary, two or more sub-layers with a common and minimal functionally to allow interface operation with the adjacent layers; and
13. Permit by-passing of sub-layers.

These principles, when applied to the problem of interconnection of open systems, lead to an identification of the seven layers. The OSI environment must permit the use of a variety of physical media and of different control procedures. Principles 3, 5 and 8, therefore, suggest the use of a separate Physical layer as the bottom layer in the seven-layer OSI architecture. The Physical layer enables a user entity to transmit or receive a sequence of bits using an encoding scheme that is most suited for the particular communication media.

Each physical media, such as telephone lines, offers a different set of data transmission characteristics, for example, channel capacity, bit error rate, and propagation delay. It, therefore, requires special techniques the transmit data between two neighbouring nodes in order to tolerate high error rates, or to take advantage of long propagation delays, as in the case of satellite channel. Similarly, reliable media, such as fibre-optic cables, require data link control procedures that are different from those used over telephone lines or satellite channels. Different techniques for data link control have been developed and user over a variety of physical communication media. Application of principles 3, 5 and 8, above, suggest the use of a separate Data Link layer on top of a Physical layer.

In an open system the topology for system interconnection may be quite different, and may, therefore, require that some systems act as intermediate relay nodes while others act as final source and destination of data. As a consequence, and using principles 3, 5 and 7, the use of a Network layer on top of the Data Link layer becomes necessary. This layer provides and end-to-end communication path between open systems using appropriate routing techniques and relaying.

In order to provide a reliable and efficient data transport service between computer systems a, Transport layer above the Network layer becomes essential. This is also consistent with principles 2, 5 and 6. As a result the higher layers are no longer concerned with issues relating to transportation of data across the network. Further, as suggested by principle 7, an interface corresponding to the Transport layered services may at a later date be subject to standardisation.

Clearly, there is need to organise, manage and synchronise interaction between Application entities. These functions are all related and quite different from those encountered earlier in the lower layers. Application of principles 3 and 4 results in the definition Session layer on top of the Transport layer. Similarly, issues concerning representation of user information exchanged between Application entities are clearly distinct from those addressed by other layers. A Presentation layer is, therefore, included in the OSI architecture so that an Application entity in an open system may use locally defined syntax and still be able to communicate with every other peer entity.

The main purpose of the OSI is to permit the users to implement distributed applications across a network of open systems. The Application layer provide a number of OSI services, for example association control, reliable transfer, message handling. The collection of protocols required to support such services are implemented as sub-layers of the Application layer.

3.2 Description of Layers

Below, we present a brief description of the nature of services provided by each layer and the functions required to be implemented to support the services. We distinguish those functions that are optional from those that are mandatory.

3.2.1 The Application Layer

The application layer is the highest layer of the OSI architecture, and permits application processes to access OSI capabilities. The purpose of the layer is to serve as a window between correspondent application process so that they may exchange information in the open environment. The description of the Application layer makes use of three definitions.

An Application entity is a model of those aspects of an application process that are significant from the viewpoint of accessing OSI capabilities. Each Application service element uses the underlying OSI communication services to provide a specific application-level-service, reliable transfer or message handling, for instance. Unlike services provided by the lower layers, application-related services are not provided to any higher layer and, therefore, do not have access points attached to them. Application service elements themselves use services provided by each other and by the lower, Presentation layer.

A user element is that part of an application process which models a user's application program, but only to extent that it uses services provided by the Presentation layer and the required Application service elements.

Application layer services and related protocols are classified into two groups, Common Application Service Elements (CASE) and Specific Application Service Elements (SASE). CASE elements are commonly required by user elements and by SASE elements, whereas a SASE element is included as part of an Application entity only when the application specifically requires the corresponding service. Some examples of the latter are message handling, file transfer and virtual terminal access. On the other hand, association control, reliable transfer and remote operations are common application services which typically used by SASE elements. Association control, for instance, enables its users to negotiate and establish the communication environment between Application entities. Once that is done protocol-data-units concerning user elements and Application service elements may be exchanged.

Functions implemented within the Application layer are very much dependent upon the service provided by each service element. But there are a number of functions that are commonly found in most Application layer protocols. These include:

1. Identification of communicating Application entities,
2. Determination of their access rights and user authentication,
3. Negotiation of the "abstract syntax" of Application protocol and use data,
4. The use of lower layer services, and
5. Error detection and notification.

3.2.2 The Presentation Layer

The Presentation layer is responsible for the appropriate representation of all information communicated between Application entities. It covers two aspects, the structure of user data, and its representation during transfer in the form of a sequence of bits or bytes. Note that the Presentation layer is only concerned with the syntax and its logical structure, not with the meaning given to it by Application entities.

A notation, called Abstract Syntax Notation (ASN), for defining the structure of Application protocol-data-units and of user information is available. It enables a sending entity to represent information using a syntax that is local to the open system. This syntax may differ from the one used to store the information in another system or during transfer between the systems. The main functionally the Presentation layer, therefore, is to transform information from its local representation to the one used during transfer, or vice versa. It thereby relieves Application entities from issues related to representation of information.

To support the above, the Presentation layer implements the following functions:

1. Connection establishment, and its termination,
2. Negotiation and possibly re-negotiation of the abstract syntax of Application protocol-data-units,
3. Syntax transformation including data compression, if required, and
4. Data transfer.

A number of services provided by the Session layer are also transparently made available by the Presentation layer. That is, for such services no additional functionally is built into the Presentation layer itself, except to map the service requests onto corresponding Session services.

3.2.3 The Session Layer

The main functionally of the Session layer is provide Presentation layer entities with the means to organise exchange of data over a connection either in full-duplex or half-duplex mode of communication. That is, depending upon the application, user entities may decide to take turns to transfer data. It also enables users to realise operation. In fact, the connection release may even be negotiated, in which case a user entity retains the option to reject a connection release.

Synchronisation points, when established in stream of data exchange, enable the two users to structure their communication in the form of dialogue units. It thereby enables them to resynchronise data exchange to an earlier synchronisation point. Resynchronisation may be useful in case of errors or, more generally, to reset the connection to an earlier defined environment. The Session layer also allows users to define an activity. Activities are another way of providing structure to data exchange between users. Aside from starting or ending an activity, a user may interrupt the activity in the midst of communication and later resume it.

In order to support the above services, the Session layer implements the following functions:

1. Connection establishment and its maintenance,
2. Orderly connection release, which may optionally be negotiated,
3. Normal data transfer, which may be half-duplex or full-duplex,
4. Typed data transfer, which is not subject to restrictions imposed by the half-duplex mode of communication,
5. Expedited data transfer, which is not subject to flow control restrictions,
6. Establishment of synchronisation points and resynchronisation,
7. Activity management, and
8. Reporting of exceptional conditions

3.2.4 The Transport Layer

While the Network layer, and those below, provide a path for data transfer between host computers, the Transport layer provides a facility to transfer data between Session entities in a transparent, reliable and cost-effective manner. It is the responsibility of this layer the optimise the use of Network services and ensure that the quality of Transport services is at least as good as that requested by the Session entities.

The Transport layer protocol has end-to-end significance, and is therefore, implemented in host computers only. The protocol makes use of the available Network services and is, therefore, not concerned with issues of routing, etc. In view of the fact that the characteristics and performance of the Network service may vary substantially, a variety of Transport protocols are available to ensure that the service that it provides is largely independent of the underlying communication network. At one extreme, whenever the Network layer provides a reliable service, the functions implemented within the Transport layer are limited to:

1. Connection establishment and its maintenance,
2. Normal and expedited data transfer,
3. Error detection and reporting.

But, if the Network service is such that user data may be corrupted, lost, duplicated or delivered out of sequence, then the Transport layer protocol must detect errors and recover from them. Functions that are additionally implemented are:

1. Error detection and recovery,
2. End-to-end sequence control of protocol-data-units.

In order to transfer data in a cost-effective manner and to match user requirements in terms of the quality of Transport service, the Transport layer uses one or more of the following functions:

1. Multiplexing or splitting of Transport connections onto Network connections,
2. End-to-end flow control,
3. Segmentation, blocking and/or concatenation.

3.2.5 The Network Layer and Below

The basic purpose of the Network layer, and those below, is to provide data transfer capability across the communication sub-network. The required functions are, as a consequence, specific to the communication sub-network and must be implemented by each open system in the sub-network and must be implemented by each open system in the sub-network, including intermediate systems. Intermediate systems are capable of routing and relaying information between possibly dissimilar communication sub-networks. Thus, the Network layer relieves Transport layer entities from all concerns regarding sub-network topology and their interconnection, and regarding routing and relaying through one or more sub-networks.

The Network layer provides the means to establish, maintain and terminate Network connections between open systems. It specifies the functional and procedural means to transfer data between Transport data entities over a Network connection. A Network connection may involve messages to be stored and later forwarded through several communication sub-networks. In order the suitably relay user data from the source host to the destination computer through one or more sub-networks, a route must be determined either centrally or in a distributed manner. Messages must also be routed within each sub-network through which the connection is established.

The major set of functions required to be implemented by a connection-oriented Network layer protocol includes:

1. Connection establishment and its maintenance,
2. Multiplexing and possibly splitting,
3. Re-initialisation, or reset, of connection,
4. Addressing, routing and relaying,
5. Normal and flow control,
6. Sequencing and flow control,
7. Error detection, notification and possibly recovery.

Alternatively, the Network layer may provide connection-less data transfer service, in which case the only significant set of functions built into the Network layer is data transfer, routing and relaying. Segmentation may also be used to ensure that Network protocol-data-units can be accommodated within buffers maintained by the Data Link layer. The purpose of the Data Link layer is to provide functional and procedural means to establish, maintain and release connections between Network entities and to transfer user data. This layer is also responsible for detection and possible correction of errors occurring over the Physical connection. Connection-oriented Data Link services are supported by the following functions:

1. Connection establishment and release,
2. Splitting of Data Link connections,
3. Delimiting and synchronisation of protocol-data-units,
4. Error detection and recovery,
5. Flow control and sequenced delivery.

Alternatively, a Data Link layer may simply support connection-less data transfer capability. In that case each service-data-unit is transferred independently of all other service-data-units. Such a Data Link layer requires a minimal set of functions to be implemented.

The Physical layer provides mechanical, electrical, functional and procedural means to establish, maintain and release physical connections and for bit transmission over a physical medium. The services provided to the Data Link entities include connection establishment and in-sequence transmission of bits over a data circuit. The Physical layer may, alternatively, provide connection-less data transfer capability, as in the case of local area networks.

3.3 OSI Layer Standards

While the Basic Reference Model discusses the OSI architecture in its totality, the detailed development of each layer in the architecture requires a careful study of solutions to fundamental problem posed for each layer. The outcome of each study takes the form of service and protocol standards for a layer. These standards are previewed in this section.

A number of organisations, in particular CCITT, ISO, IEEE and ECMA, have been developing standards for the six bottom layers, Physical layer through Presentation layer, and for different application of the Application layer. These organisations work independently, but have co-operated with each other by adopting many of each other's standards as their own. This has not only cut down the time and cost of development of OSI standards, but has led to the development of a consistent and compatible set of standards for the seven layers.

For the layers, Physical layer through Presentation layer, a standard typically consist of two documents, one of service definition while other covers protocol specification. These documents may make references to one or more related documents as well. Standards for Transport, Session and Presentation layers literately fit into this document structure. The situation regarding the lower layers is different since several options are available a designer regarding the choice of communication media, sub-network topology and their interconnection, less. Note that, particularly because of options concerning Network service and protocol, multiple Transport layer protocols may have to be defined so that the Transport service is uniformly identical across the entire network.

Application layer standards are also differently documented. There may be several documents concerning a single application. Two of these documents, again, relate to service definition and to protocol specification. The other documents discuss related concepts and the application itself. In some cases, such as message handling, more than one services may be defined to cater to a variety of users or equipment.

Layer	Service Documents	Protocol Documents	Other Documents
Application layer			9545
Association control	8649	8650	-
Reliable transfer	9066-1	9066-2	-
Remote operations	9072-1	9072-2	-
CCR	9804	9805	-
Directory services	9594/3	9594/5	9594/1,/2,/4,/6,/7
Message handling	10021-4, -5	10021-6	10021-1, -2, -3, -7
File transfer	8571/3	8571/4	8571/1, /2
Virtual terminal	9040	9041	9646, 2022
Presentation layer	8822	8823	8824. 8825
Session layer	8326	8327	-
Transport layer	8072	8073, 8602	-
Network layer	8348	8878, 8473	8208, 8648, 8880, 8881, 9068, 9542
Data Link	8886	8802	7776
Physical layer	10022	-	8802

Table 3.1 ISO Documents Pertaining to Each Layer (Documents are Numbered as, for Example, ISO 9545)

Tables 3.1 summarise the related standards documents, for each layer from ISO and CCITT. For most layers, ISO and CCITT standards are identical except for editorial changes. Such standards are termed as co-standards. The co-standards, in general, permit interoperability between implementation conforming to ISO and CCITT standards.

Similarly, Figure 3.2, describe the constrains on the use of higher layer protocol in conjunction with a lower layer service [CCITT X.220]. These constraints are, in fact, part of the protocol specification since each protocol must explicitly state the service it expects of the supporting lower layer.

It may be pointed out that not all the standards documents, referred to above, are accepted as standards by ISO. Some of the documents may still be at the stage of Draft International Standard (DIS) or even Draft Proposal (DP). These documents are, therefore, subject to minor changes, if not major ones.

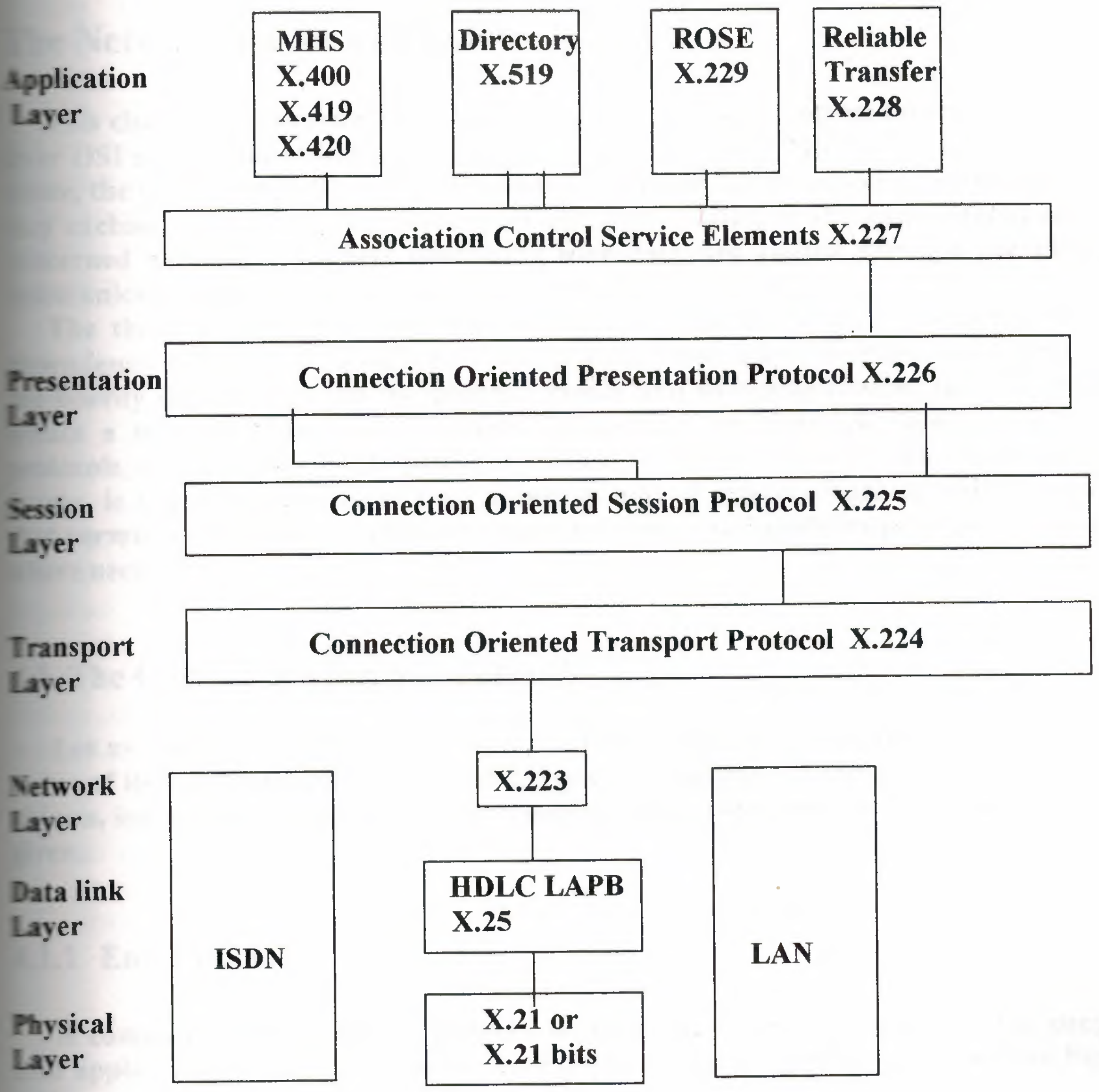


Figure 3.2 Standards recommended by CCITT.

HAPTER 4

he Network Layer and Below

This chapter is concerned with a discussion of the three bottom layers of the seven-er OSI architecture, that is the Network, Data Link, and Physical layers. Taken as a ole, the three layers offer to entities in the Transport layer a service using which they exchange user data. The Transport entities, residing in the end systems, are not cerned as to how packets containing user data are routed through the physical munication network, and how such a network is accessed.

The three layers are treated together in one chapter in order to bring out the endency of the corresponding protocols upon each other. Further, these protocols heavily dependent upon the physical media and switching/routing techniques used thin a network. The emphasis here, however, is on network layer services and tocols, and how they relate to protocols used to access a real network. Also contained rein, is a discussion of local area networks and of internet-working using gateways at permit interconnection of two or more networks, and perform protocol conversion, here necessary.

1 The Communication Sub-network

Let us consider the physical structure of the computer communication network in ms of its functional components first. Formal definitions of terms used to model real jects, including an end system, a communication sub-network and a relay system are ven.

.1.1 End Systems

A computer communication network is collection of real end systems, that support er application process, and one or more physical communication networks (see Figure 1).

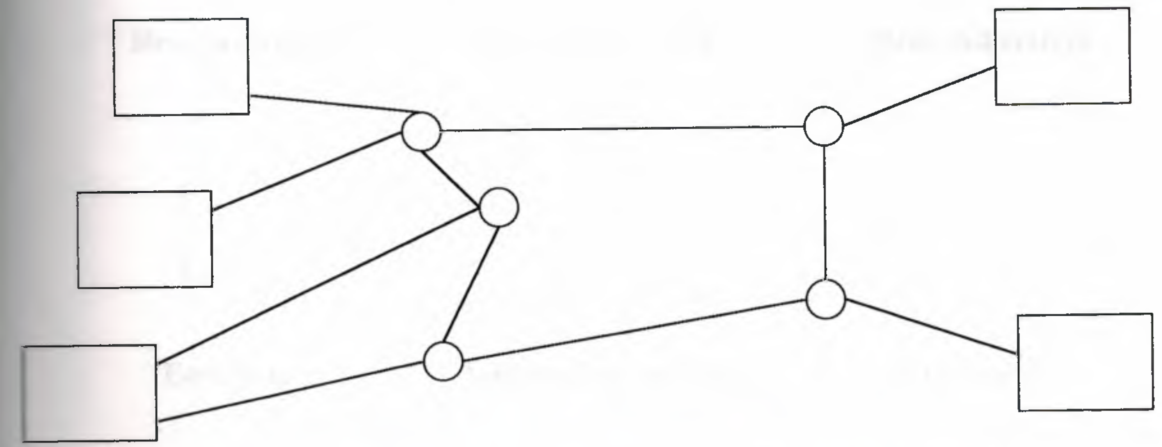
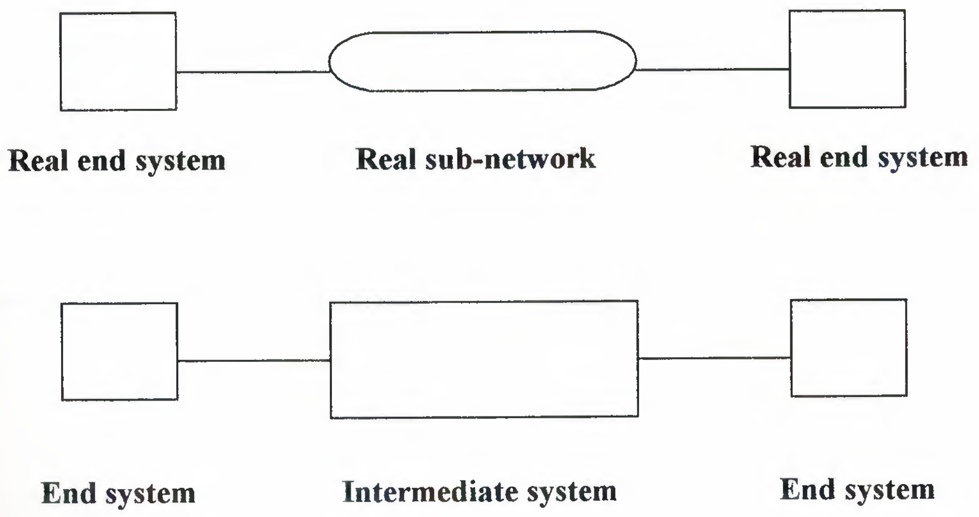


Figure 4.1 A computer communication network.

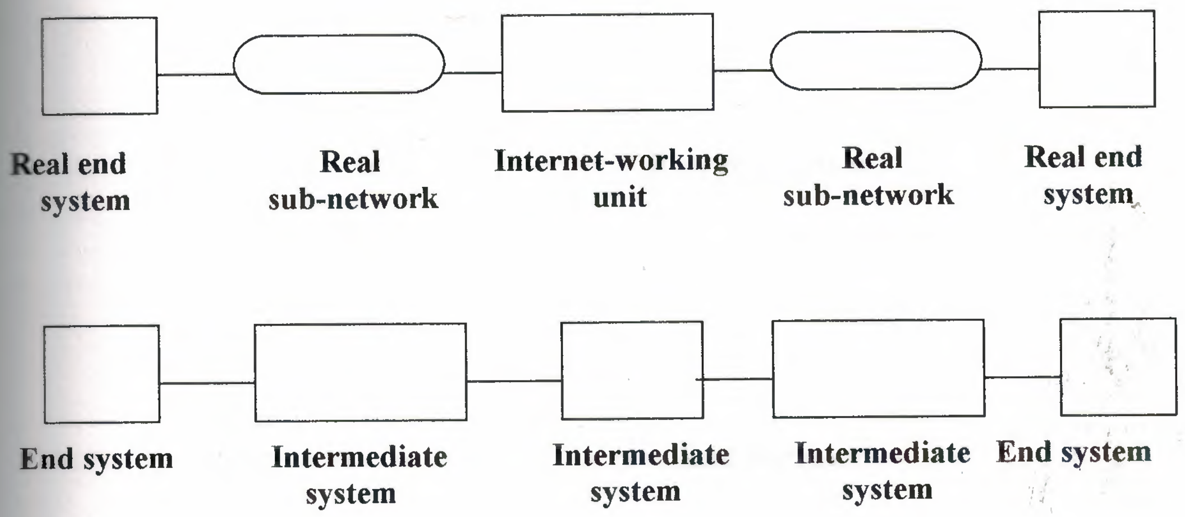
A real end system transfers data to other real end systems using the data transfer facility provided by the communication network. This interaction with the communication is governed by a protocol that is specified to the interface between the real end system and the communication network. From the viewpoint of the OSI architecture, only those aspects of the real end system that concern communication with the network or with other real end systems are of interest. This abstraction of the real end system will simply be referred to as an end system, or ES. Formally, an end system is an abstraction of a system that hosts user applications. Such a model includes the protocol that it uses the access the communication network, and protocols that concern communication with other end systems.

4.1.2 Sub-networks

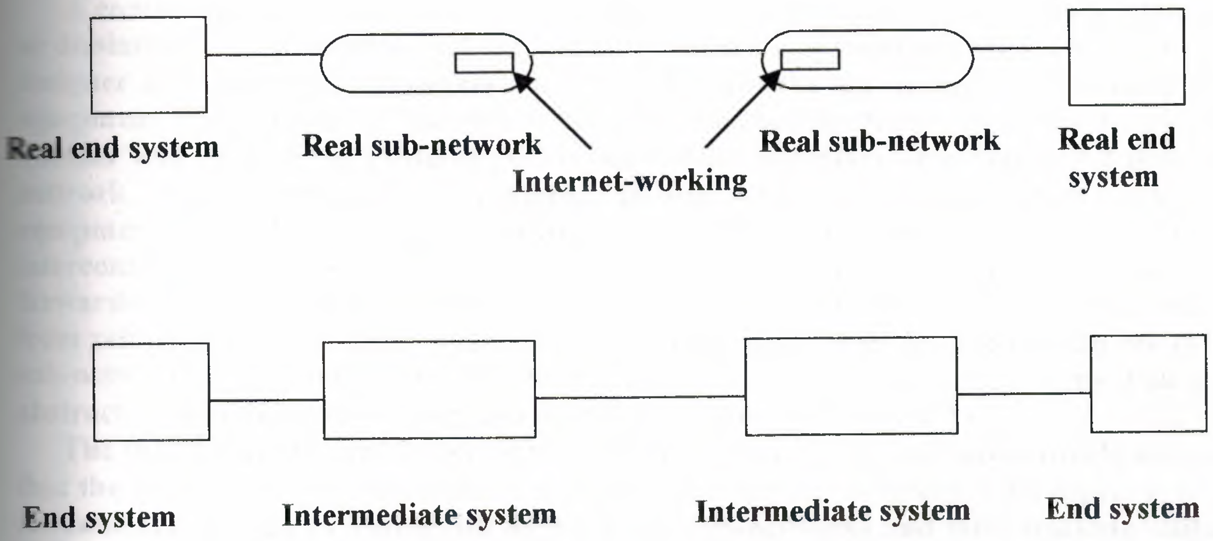
The physical communication network is a collection of equipment and physical media viewed as one autonomous whole that interconnects two or more real end systems. Such a network may be a public or a private network. Further, it may be a wide area or local area network. The term real sub-network is used to denote the physical communication network. From the viewpoint of the design of the Network layer, the internal working of the real sub-network is unimportant. What is of significance is the interface that it offers to end systems so that information may be exchanged across the network. Using this interface an end system may access sub-network resources to establish connections or to simply transfer information to other end systems. Thus, from the viewpoint of OSI architecture, the entire real sub-network can be viewed as one whole without concern for its internal details. In OSI terminology, the real sub-network is simply referred to as a sub-network. Figure 4.2(a) illustrates this. At the very least, the sub-network enables an end system to transfer data across the sub-network to another end system.



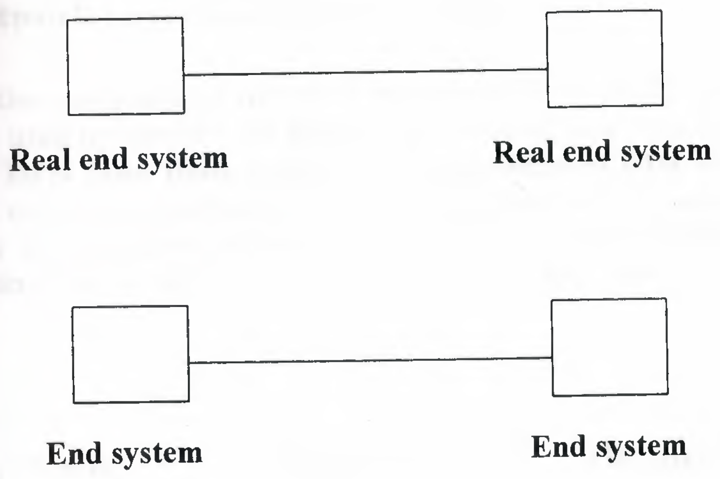
(a) A network consisting of a single sub-network.



(b) A network consisting of two sub-networks connected using a relay system.



(b) A network of sub-networks, each implementing different access protocols at its interfaces.



(c) A network where end systems are directly connected.

4.1.3 Inter-working

A communication network may be formed by interconnecting two or more similar, or dissimilar, communication sub-networks. From the viewpoint of OSI architecture, a designer may view the interconnected. Two sub-networks are interconnected using an equipment whose primary function is to relay information from one sub-network to another and to perform protocol conversion, where necessary (see Figure 4.2 (b)). A network layer gateway is one example. In the world of communication such an equipment is called an Inter-working Unit (IWU). Obviously, when an IWU interconnects two sub-networks so that data, received over a sub-network, can be forwarded to an system connected to the sub-network (or to another IWU). Thus, aside from relaying, an IWU must perform protocol conversion if its interfaces with the two sub-networks are different. In OSI terminology, the term relay system is used to be abstract the functions of relaying and of protocol conversion in an IWU.

The function performed by an IWU is similar to that of any real sub-network, except that the protocols at the two interfaces of the IWU may be different. This aspect is not fundamental enough to distinguish between real sub-networks and inter-working units. A sub-network may also offer different interfaces to host or to other sub-networks, depending their communication requirements (see Figure 5.2 (c)). Thus, from the viewpoint of OSI architecture, real sub-networks and IWU are treated alike, and are also referred to as Intermediate Systems. Formally, an Intermediate System (IS) is an abstraction of equipment and/or communication media which performs the function of relaying (and routing) of information to end systems or the other intermediate systems. This abstraction takes the form of an access protocol specified for each interface. From this perspective, an end system does not perform any relay functions.

Finally, two real end systems may be directly connected using a communication link, or possibly through a shared media. The network and its model are illustrated in Figure 4.2 (d). The important point to be noted is that a sub-network based on a shared medium does not perform a relay function within the sub-network.

4.2 The Network Layer and Below: A Model

In this section we discuss a model of the three bottom layers of the OSI architecture. This model is used to describe the functions of routing and relaying of user data through the Network layer (and those below). Characteristics of data transfer, irrespective of whether it is connection-oriented or connection-less, are discussed. Further, we take a close look at the structure of an Intermediate system to highlight the relationship between routing and relaying and the three layers of protocols.

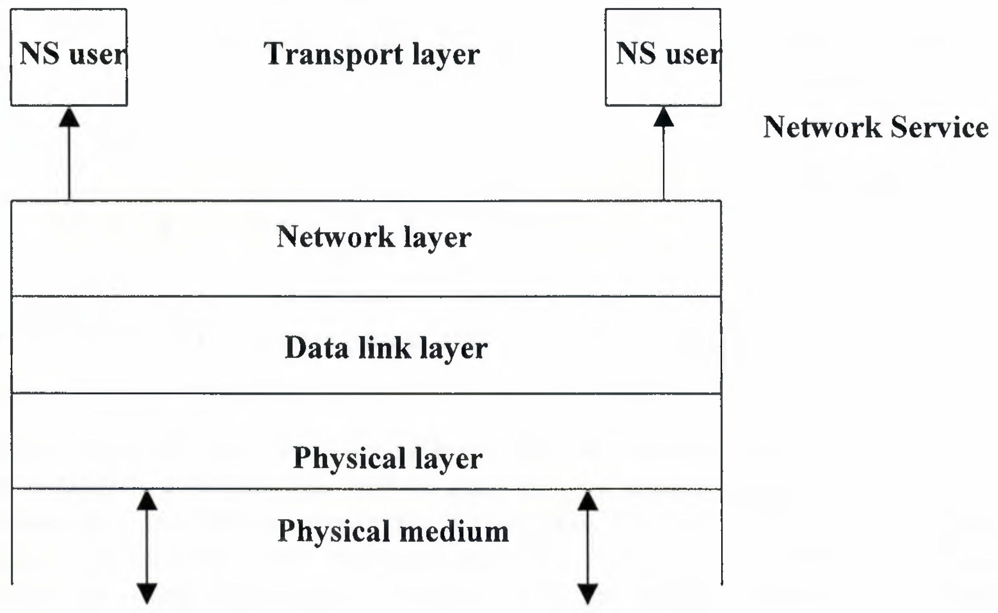


Figure 4.3 The Network Service provider and its users.

4.2.1 User-Provider Model of Network Service

A model of the Network layer together with the two bottom layers, the Data Link and Physical layers, is given in Figure 4.3. The Network layer offers to entities in the Transport layer a capability by which they may exchange data across the physical sub-network, without concern for how it is actually routed or relayed through the sub-network. entities within the Network layer and those below co-ordinate their operations the provide a service, called Network Service (NS). Together, the Network entities (and those below the Network layer) are modelled as the Network Service Provider (NS Provider). Since the next higher layer is the Transport layer, the users of the Network service are Transport entities residing in end system. As such, a Transport entity is called a Network Service User (NS User). Transport entities what wish to use the Network service are bound to one or more NSAPs, as shown in Figure 4.4.

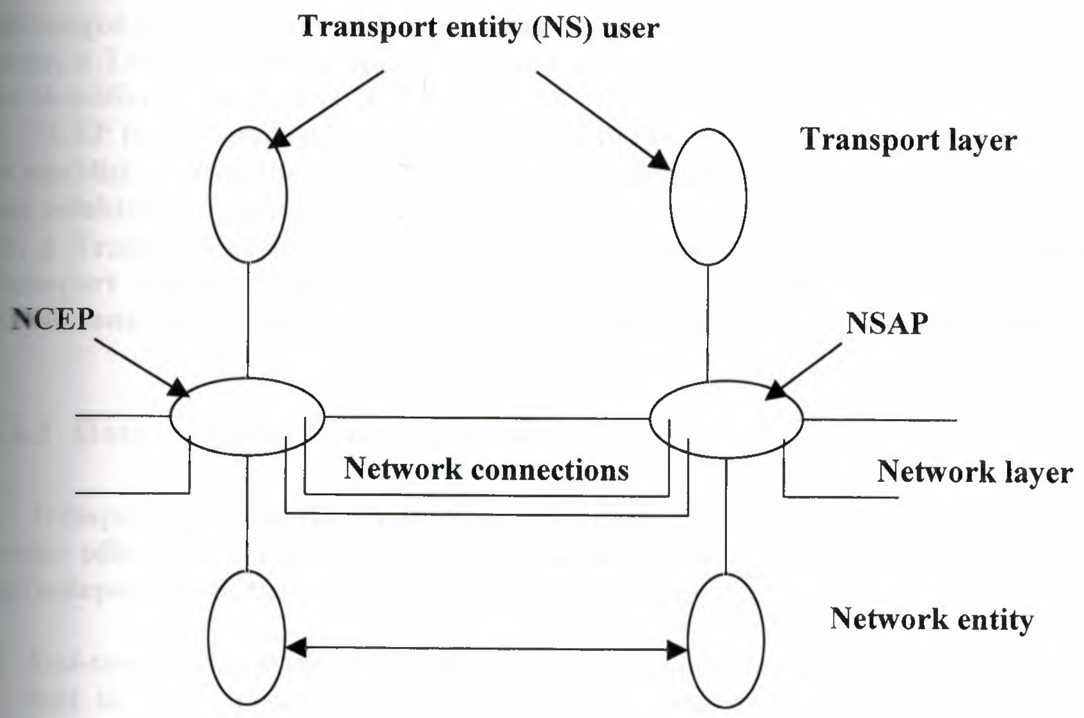


Figure 4.4 Network service access points and connection end points.

The network Service may be connection-oriented, in which case two Transport entities must establish a connection before data can be transferred. The connection is preferably released soon after data transfer is complete. Or, the Network service may be connection-less. In connection-less Network service, a Transport entity simply makes available user data to the NS provider, together with the address of an NSAP to which the destination Transport entity is attached. The NS provider than appropriately routes packets through the sub-networks to the destination NSAP. A packet, or more precisely a Network Protocol Data Unit, consists of user data and the addresses of both the source NSAP and destination NSAP. The source NSAP address may be used by the destination Transport entity to determine the identify of the source Transport entity. These addresses constitute only a part of the packet header, better known as Protocol Control Information.

4.2.2 Network Connections

A similar approach is used by the Network layer to establish a connection on behalf of a pair of Transport entities (see Figure 4.4). Such a connection is formally termed Network Connection (or NC). Once a connection is established, packets containing user data do not explicitly carry the address of source and destination NSAPs. Instead, they carry information that uniquely identifiers the connection to which the data belongs. From the viewpoint of view of Transport entities, a connection is identified by its end points, one in each NSAP. These are formally referred to as Network Connection End Points, or NCEPs, and are identified using, what are formally termed, Network Connection End Point Identifiers (NCEP Identifiers). Each NCEP identifier has a local significance, in that it is unique within the domain of the corresponding NSAP. The NCEP Identifiers are assigned at the time of connection establishment, and remain unchanged during the lifetime of the connection. Subsequently, during the data transfer phase, a Transport entity makes available user data to the NS provider, together with the identifier of the local NCEP.

NCEP Identifiers also help to distinguish between a number of connections that may be established from the same NSAP. This is required since a pair of Transport entities may establish a number of connections to support transfer of unrelated streams of data. Or, a Transport entity may establish an independent connection with a number of Transport entities. The NS provider is invariably capable of supporting multiple connections, subject of course to availability resources, primarily storage related.

4.2.3 Data Transfer Characteristics

Irrespective of whether data transfer is connection-oriented or connection-less, the service offered to a Transport entity is characterised as being end-to-end, transparent, and independent of the underlying communication media.

1. *End-to-end data transfer* is made possible through the use of intermediate system, that is, sub-networks and/or relay systems, which together are responsible for appropriately relaying and routing user data through the network and delivering it to the destination end system. The address of destination end system is provided by the source end system. Addressing is one of the important issues concerning the Network layer and shall be discussed later in this chapter.
2. *Transparency of data transfer* refers to the fact that the Network layer (and those below) do not place any constraint on the contents of user data, since user data is not interpreted by the Network layer.
3. *Independence* from the underlying media implies that NS users are neither concerned with the characteristics of the underlying transmission media, or with the protocol used to access the sub-networks. It is a different matter that the decision the provide a Network service, which is connection-oriented or simply connection-less, may depend upon the communication media and sub-network protocol. Further, the quality of the Network service is, to large extent, dependent upon the sub-network.

4.2.4 Intermediate Systems: A Model

Consider the simpler case where en systems are connected using an Intermediate system which offers identical access to all end systems. This situation, earlier modelled in Figure 4.2(a), is elaborated in Figure 4.5 to show the three distinct layers, and the context of routing and relay functions. The end systems have entities in the Transport layer and those above. An Intermediate system, on the other hand, aside from implementing protocols at the three layers, also implements routing and relay functions. As is clear from the figure, user data received from an end system over an interface is processed by the sub-network and relayed another interface. It is subsequently forwarded to the destination end system.

The upper triangle in the Intermediate system, Routing and Relaying, is more than a simple software module. It is, in fact, a representation of the entire physical sub-network consisting of switching nodes, transmission media, and protocols that are totally internal to the sub-network. From the viewpoint of the OSI architecture, it is adequate to abstract the sub-network as simply implementing a routing and relay function, and providing interfaces to end systems.

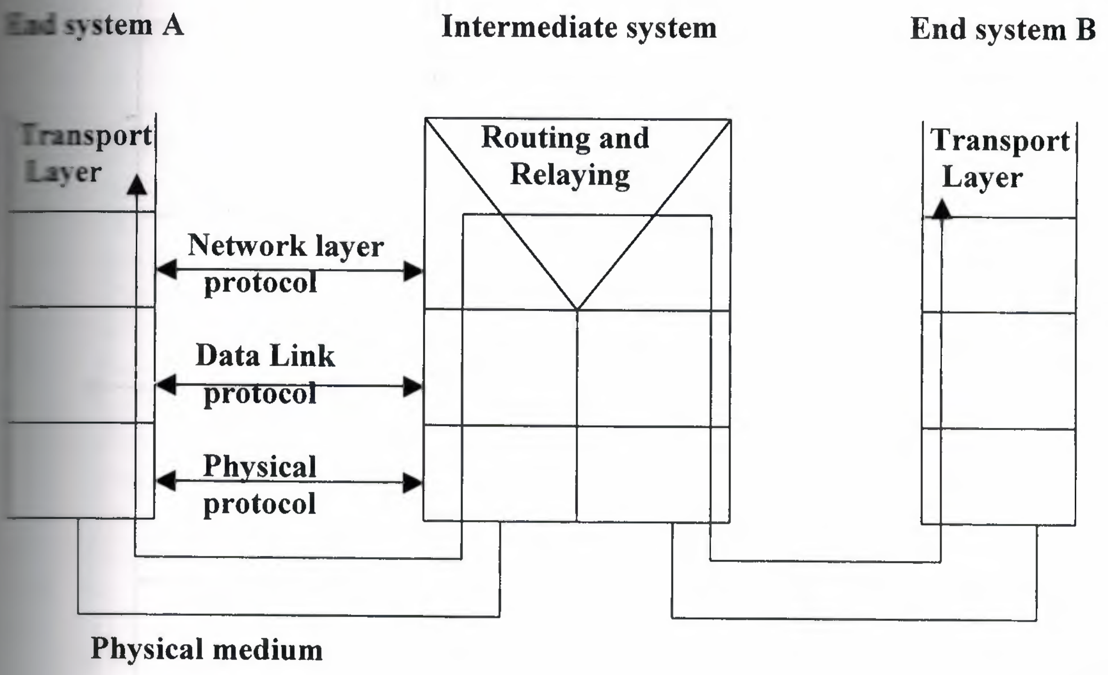


Figure 4.5 Elaboration of the structure of an Intermediate System.

The design of the Intermediate system can be further elaborated, as in Figure 4.6. The two network layer entities within an Intermediate system interface with each other using a protocol, the specification of which is outside the scope of the OSI environment.

Note that Network layer entities, within an Intermediate system, there are neither NSAPs, or Network connection end points. Thus, a Network connection, logically, extends from an NCEP, in an NSAP in an end system, to an NCEP in another end system. As discussed earlier in this section, communicating Network entities identify a connection using a Network Protocol Connection Identifier which has local significance only. Thus, one of the functions implemented by Networks entities in Intermediate systems is to maintain a correspondence between the two identifiers used to identify the same connection, but over the two interfaces.

A Network entity, whether it is an end system or an Intermediate system, uses Data Link services made available at a Data Link Service Access Point (or DLSAP). This it does to transfer Network Protocol Data Units across the interface to a corresponding Network entity. Similarly, a pair of Data Link entities, one each in an end system and an Intermediate system, uses Physical layer services made available at Physical Service Access Points (or PhSAPs).

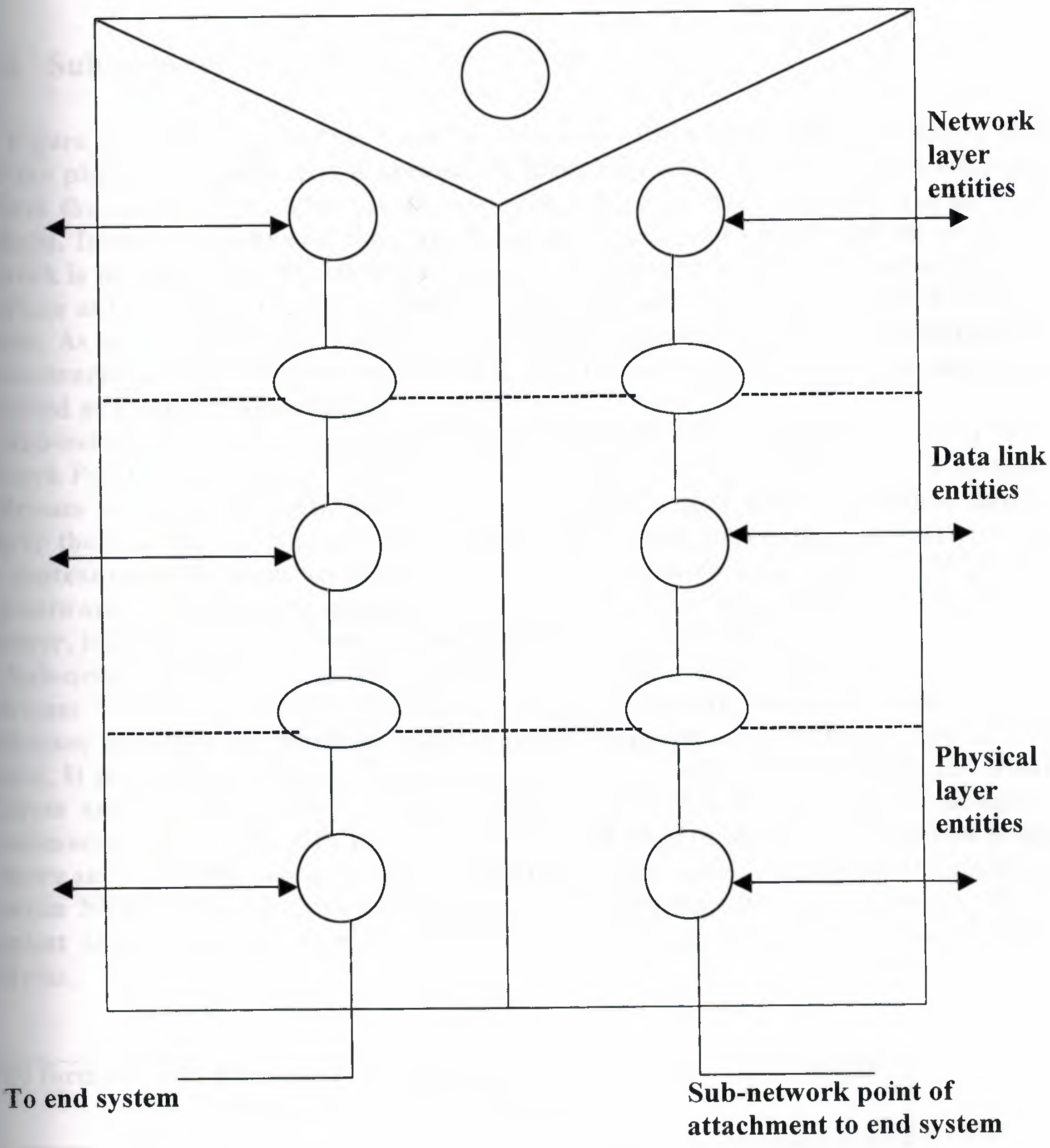


Figure 4.6 Entities within an Intermediate system and their interfaces.

4.2.5 Sub-network Access Protocol

The protocols at the three bottom layers specify the procedure that should be used by entities in end systems to access the routing and relaying capability of sub-network. this collection of protocols is also called Sub-network Access Protocol or simply SNAcP. The SNAcP may, of course, vary from one sub-network the another. For example, CCITT's X.25 protocol specifies the interface a host computer must use to establish a network layer connection with another host. This connection is routed through a packet-switched sub-network. Similarly, a local area network provides connection-less or connection-oriented data transfer between end systems over a shared communication channel.

4.2.6 Sub-network Addresses

Figure 4.6 also brings out the interface between a Physical entity, in an end system, and the physical medium that it accesses to transmit/receive a stream of encoded bits to form the sub-network. There is no service access point associated with the physical medium. Instead, the physical interface between the real end system and the real sub-network is modelled as a Sub-network Point of Attachment (or SNPA). It is the physical interface at which data transfer capability of the sub-network is available to a real end system. As an example, a host computer may physically interface with a communication sub-network though a modem connected to one of its RS 232C ports. The modem is modelled as a point of attachment.

Sub-network Points of Attachments are identified by an address, termed Sub-network Point of Attachment Address, or Sub-network Address simplicity. Sub-network Addresses are used to route Network layer packets within the sub-network and to deliver them to the appropriate end system, relay system or another sub-network. In the context of public data networks, a DTE Address is used to identify an SNPA. The Sub-network Addresses are assigned by the sub-network administration. Their scope, however, is local to sub-network and the outside the OSI environment.

Sub-network addresses must be distinguish from NSAP Addresses. The NSAP addresses are used to identify Network service access points, whereas a Sub-network Addresses identifies the physical interconnection between a sub-network and an end system. It is a different matter that there may be a correspondence between the NSAP Address and a Sub-network Address. Such a mapping must be flexible enough to accommodate multiple NSAPs in an end system and attachment of an end system to one or more sub-networks using a number of physical links. Thus, in general, the mapping between NSAP addresses and Sub-network Addresses may be many-to-many. In the simplest of cases an NSAP address may be mapped one-to-one onto a Sub-network Address.

IDI format	AFI (decimal value)	IDI (no. of decimal digits)	DSP (max. length)
X.121	36 37	14 14	24 digits 9 octets
ISO DCC	38 39	3 3	35 digits 14 octets
Local	48 49 50 51	0 0 0 0	38 digits 15 octets 19 ISO 646 char. 7 National char.

Table 4.1 Assignment of NSAP Addresses

While assignment of NSAP addresses may be done in one of several ways, a hierarchical structure permits interoperability between open systems, while retaining control over the assignment of addresses by local administrations. This structure is briefly described below.

The ISO document [ISO 8348 DAD 2] specifies, at the highest level, that an address may conform to any one of the several available standards, including CCITT's X.121, ISO DCC and LOCAL. At the next level, X.121 for example, may specify the country codes and perhaps a network identifier. The latter may be viewed as identifying the Initial Domain within which the remaining part of the NSAP address the unique. At lower levels, the assignment of addresses is domain specific, and may be specified in terms of location, machine etc. Thus, each NSAP address composed of at least three parts (see also Table 4.1).

1. The Address and Format Identifier (AFI) specifies the structure of the address. More specifically, the two AFI digits encode the particular standard that is used (X.121, ISO DCC, local, etc.) and the format (octets, digits, ISO 646 or National characters) of the domain specific part of the address.
2. Initial Domain Identifier, or IDI, specifies the initial domain within which the NSAP address must be unique. When X.121 is used, for example, the 14 digit IDI field specifies the country code (3 digits), the network number (1 digit) and the 10 digit DTE address. If a local addressing scheme is used, the IDI is absent. The 3 digit IDI in an ISO DCC specification is a country code.
3. Assignment of the Domain Specific Part, or DSP, is a local matter, as long as they are unique. The DSP is sequence of either octets or digits. (see Table 4.1).

We give below two examples, confirming to X.121 and Local:

1. Using X.121 specification

   AFI: 36 (X.121, Decimal)
   IDI: 310 5 1234567890 (USA, 5th Network, Machine Number)
   DSP: 1234567890123456 (Access Point)

2. Using Local Specification

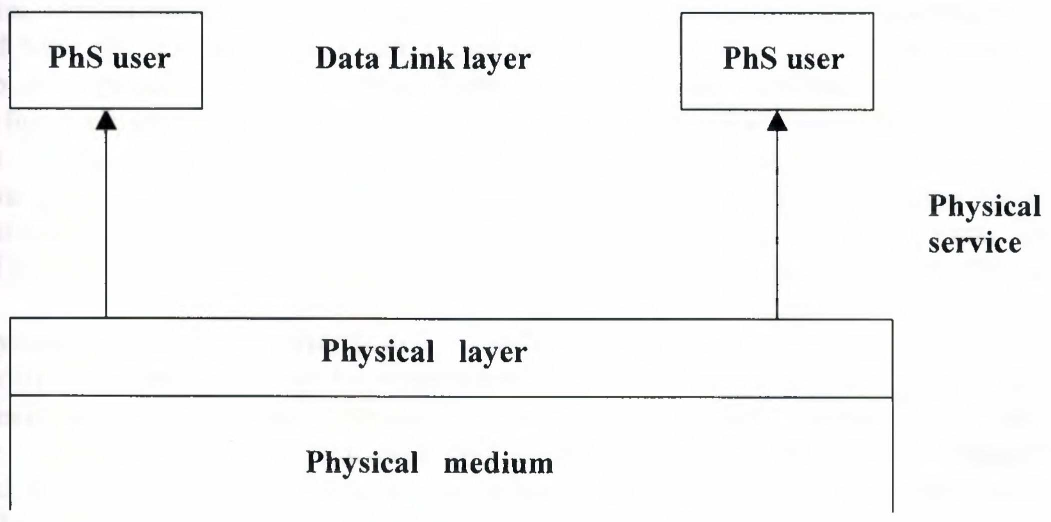
   AFI: 50 (Local, ISO646 Characters)
   DSP: EDU&EnR&IITD005&abc (Education, Network Name, Location, SAP)

4.3 Physical Layer Services and Protocols

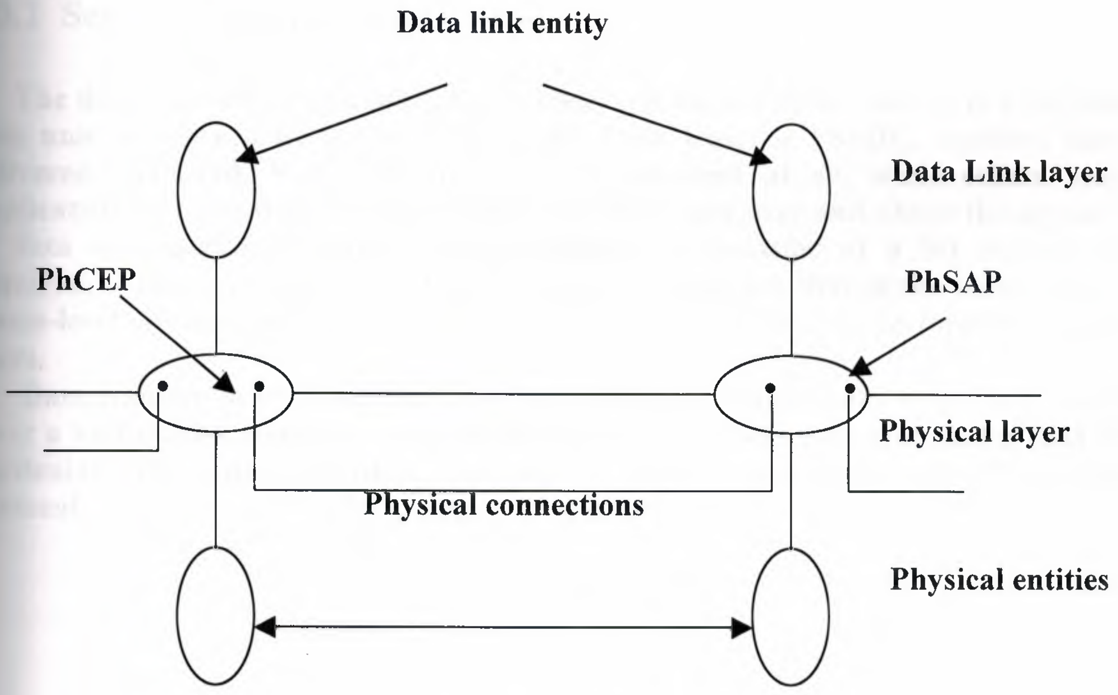
In this section we discuss the service offered by the Physical layer to Data Link entities. *Protocols that use an available transmission facility in order to support a* physical connection covered using example protocols, including RS 232C, X.21, and IEEE 802.3.

4.3.1 A Model of the Physical Layer

A model of the Physical layer is presented in Figure 4.7. It brings out the fact that the Physical layer service, formally termed Physical service, or PhS, is used by Data Link entities, irrespective of whether they reside in end systems or in Intermediate systems. As such, Data Link entities are also referred to as Physical service users, or PhS users.



(a) Physical service users and providers.



(b) PhSAPs and Physical layer.

Physical services are provided to its users at service access points, termed Physical Service Access Points, or PhSAPs.

The Physical layer provides to its users an ability to send or receive a stream of bits over a physical transmission medium. Physical service users are not concerned with how a bit is encoded in the form of electrical (or optical) signal before transmission. As such, the design of Data Link protocols can be carried out without concern for issues like modulation, voltage levels, or with pin-level description of a physical attachment to the transmission medium.

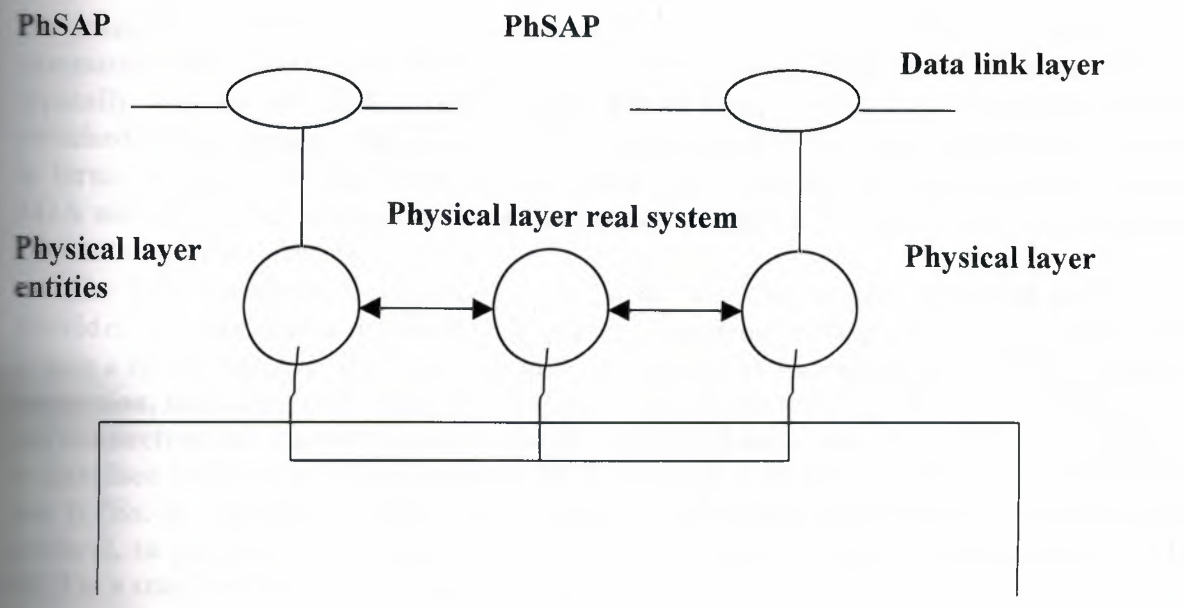
The other capability provider by the Physical layer has to do with activating or deactivating a connection. This is no different from establishing or releasing a connection. Activation of connection ensures that if a user initiates the transmission of a stream of bits, the receiver at the other end is ready to receive them. The process of activation may require that resources, both processing and transmission related, be reserved for exclusive use by this connection. Deactivation of a connection releases all resources for use by other Physical connections. Figure 4.8 illustrates how a physical connection is mapped onto a communication path or data circuit. In some cases the path may consist of more than one data circuit, interconnected using a Physical layer relay system. The relay system or its operation is not visible to the communicating Physical entities.

A Physical connection is established on behalf of PhS users between two PhSAPs. Since multiple connections may be established by a PhS user from the same PhSAP, each connection is identified by a Physical Connection End-Point Identifier, or a PhCEP identifier. There is one PhCEP for each end of the connection. The PhCEP identifiers associated with the two end points of the same connection may be (or may not be) distinct. This is so since the significance of a PhCEP identifier is local to a PhS user and the service provider. Within the Physical layer, each connection is mapped onto one physical transmission medium. Multiplexing of connections, if any, is done either within the communication sub-network.

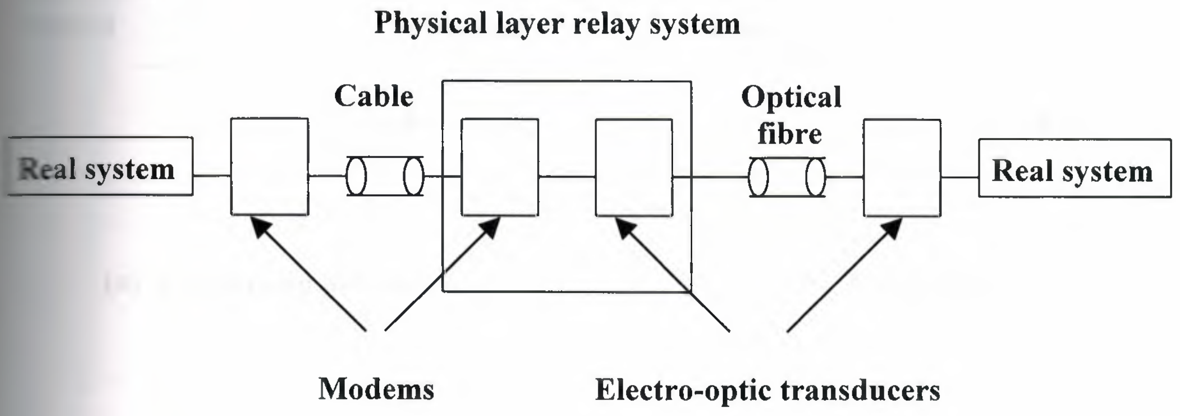
4.3.2 Service Characteristics

The data unit whose boundaries are preserved during data transfer is a bit. Such a data unit is referred to as Physical Service Data Unit, or PhSDU. Further, bits are delivered unaltered. Some bits may not be delivered at all, while others may be duplicated. Flow control may be exercised by PhS users over and above the agreed rate of data transmission. Further, synchronisation of transfer of a bit stream, when required, is the responsibility of the Physical layer and not that of the users. However, frame-level or character-level synchronisation, where required, is performed by the PhS users.

Data transfer over connection may be full-duplex, half-duplex or perhaps simplex. Over a half-duplex connection, which of the two PhS users may transmit a data bit at particular time is determined by the users themselves, and not by the Physical layer protocol.



(a) Logical view of relay operation.



(b) Real view of relay operation.

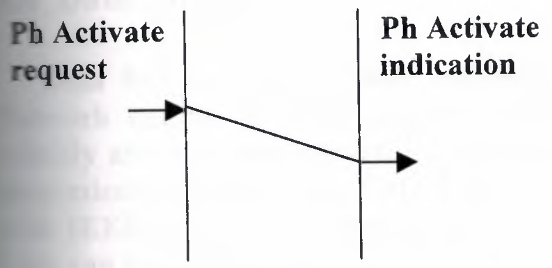
Figure 4.8 Physical connection using a data circuit through a Physical layer relay system.

4.3.3 Physical Layer Protocols

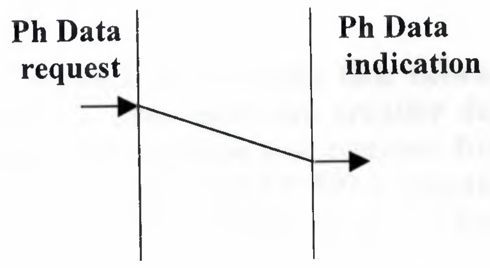
The protocols considered here are EIA's RS 232C and CCITT's Recommendation X.21. These protocols specify standard interfaces for connecting a host/terminal equipment, also called DTE, to a communication sub-network. These interfaces provide for serial base-band communication and do not permit multiplexing. The IEEE 802.3 standard is also covered briefly.

A comparison of some of the characteristics of the three protocols is given in Table 4.2. These characteristics reflect in some ways the different contexts in which these protocols are applicable. RS 232C, one of the oldest and most popular interfaces, is particularly suited for connection-oriented, character mode, asynchronous communication over short distances at relatively low data rates. A data terminal typically uses an RS 232C interface with a host computer or to a modem to access a switched/leased circuit. The greatest disadvantage of RS 232C interface is its limitation in terms of speed and distances. A subsequent EIA standard RS 449, together with RS 442A and 423A, has overcome these limitations at the cost of increasing the number of connector pins or circuits.

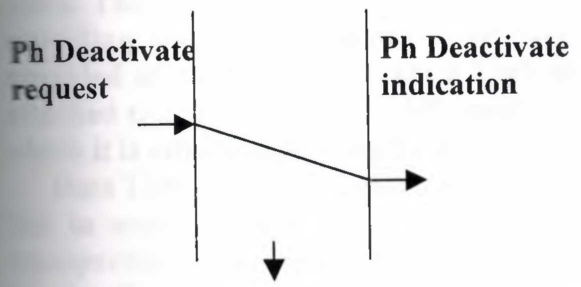
The X.21 interface is particularly suited for use over digital switched networks. It provides for connection-oriented, full-duplex, synchronised transfer of a stream of bits. It uses a fewer number of circuit (up to 8), but provides extensive control of the physical connection, including transmission of address information to the DCE so that an end-to-end connection can be established through a switched network. As such, it could be used to interface with a local DCE, remote DCE through a switched network, or to connect two DTEs. In the latter case, it may be used, in conjunction with a Network layer protocol, to provide Network layer services to Transport entities in end systems. IEEE 802.3 is a standard which covers more than just the Physical layer.



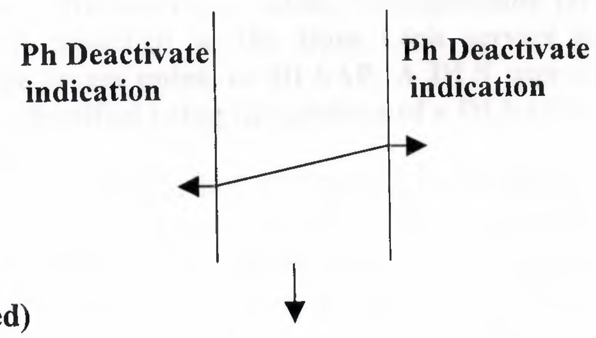
(a) Connection activation.



(b) Data transfer.



(c) Connection deactivation (user initiated)



(d) Connection deactivation(provider initiated)

Table 4.2 Characteristics of Some Physical Layer Protocols

Characteristics	RS 232C	X.21	IEEE 802.3
Connection oriented	Yes	Yes	No
Date rate	Up to 19.2kbps	Up to 9600 bps	10 Mbps
Synchronous	Asynchronous (usually)	Synchronous	Synchronous
Duplex	Full-duplex	Full-duplex	Half-duplex (Shared medium)
Encoding scheme	NRZ	NRZ	Manchester
Connector	25	15	15
No. of circuit used	9 to 16	6 to 8	9 or 10

4.4 Data Link Service

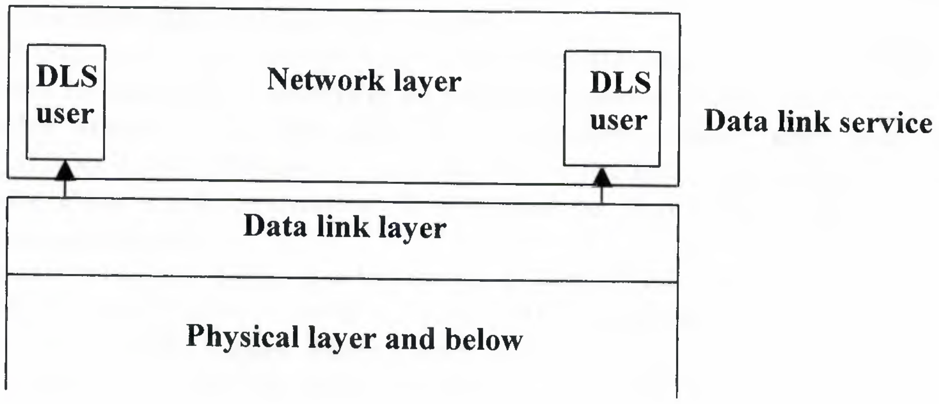
Data link services include establishment and maintenance of data link between Network entities in neigh-boring systems. Over such a link, users can transfer data reliably and without concern for framing, addressing, and detection and recovery from transmission errors. Two Data Link protocols, X.25 LAPB and IEEE 802.2 (together with IEEE 802.3), are discussed in later sections, with a view to illustrate how a Data Link can be maintained over a Physical connection.

4.4.1 A Model of Data Link Layer

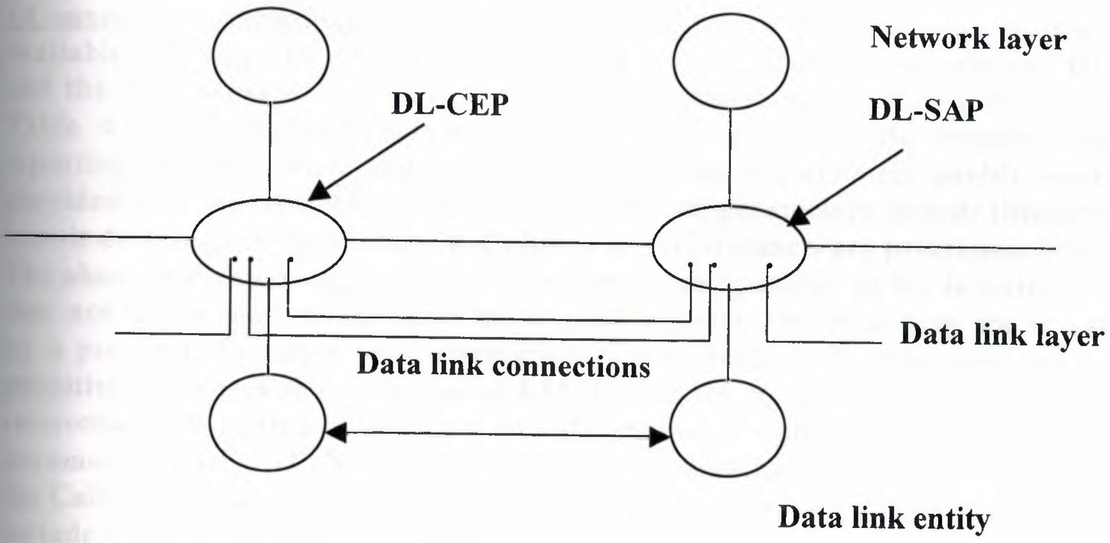
Figure 4.10 is a model of Data Link layer and the service that it offers to its users. Users of the Data Link service are Network entities residing in end systems or in Intermediate systems. They are, as such, referred to as Data Link service users, or DLS users. The Data Link layer, together with the Physical layer below, is responsible for providing the service. These layers are thus modelled as the Data Link service is provided at, what is called, Data Link service access point, or DLSAP. A DLS user is attached to one or more DLSAPs, and can be identified using the address of a DLSAP to which it is attached, or simply DLSAP address.

Data Link service is generally connection-oriented (see [CCITT X.212, ISO 8886.2]). But in some cases it may simply be connection-less. In either case, data transfer is transparent to the Data Link layer. That is user data is not constrained in any manner by the Data Link layer. Further, characteristics of the Physical layer and of the transmission medium are not visible to the DLS users. A Data Link Connection (or simply DLC) is established by the Data Link layer between two DL service access points on behalf of two DLS users. Each connection is identified by a DLC End-Point Identifier, the significance of which is local to the DLS user and the DLS provider. It may, however, be pointed the out that usually there is only one Data Link connection established between a pair of DLS users. This DLC is able to support a number of Network connections.

A Data Link Service Data Unit (or simply DLSDU) is a sequence of bits, bytes, transferred from one DLSAP to another with another with its two boundaries preserved. There may be a constraint on the maximum size of the DLSDU. Thus, one of the major concerns of the Data Link is to suitably delimit, and perhaps segment, DLSDUs so that these can be transferred from one DL entity to another. Other major functions of the Data Link layer include addressing, error detection and recovery, and flow control. Error recovery and flow control are meaningful only when the service is connection-oriented.



(a) Data Link service users and provider.



(c) DLSAPs, connections and DLCEPs.

Figure 4.10 A model of the Data Link layer, service access points and connection end points.

4.4.2 Data Link Service

The Data Link layer provides to its users the capability to establish or release a connection. Once a connection has been established between two DLSAPs, an attached DLS user may issue a primitive to the Data Link layer to transfer (normal) data, in the form of DLSDU, to the corresponding DLS user.

Normal data transfer may e slowed down due to flow control, exercised either within the Data Link layer or by DLS users. The Data Link layer may provide in additional mechanism by which a user may request urgent transfer of an Expedited DLSDU of a fixed maximum length. This is useful in case users wish to exchange control information outside the flow controlled normal data stream.

A connection may, conceivably, run into difficulties due to frequent transmission errors or from its inability to interpret an incoming protocol data unit. In that case the DLS provider simply resets the data link. Similarly, either user may reset the connection, if it so desires. In any case, all data over the link is discarded. The resulting status of the Data Link connection is identical to that which existed soon after connection establishment.

Over local area networks, particularly, a Data Link layer may only support connection-less data transfer. There, each DLSDU is considered to be unrelated to all others. The Data Link makes every effort to transfer it successfully, but without detecting errors, or exercising flow control. Such functionally, when required, it typically implemented by the Transport layer.

4.4.3 Service Primitives and Parameters

Table 4.3 lists the primitives concerning each service element, together with the parameters associated with each parameters.

1.<u>Connection establishment:</u> Connection establishment is confirmed service, and is available only when DLS connection-oriented. During establishment, the two DLS users and the DLS provider negotiate the use of optional service elements, if available (see Table 4.4). The optional service elements are expedited data transfer and error reporting. The DLS users and the DLS provider also negotiate the quality service to be provided over the connection. The quality of service parameters include throughput and transit delay. Other parameters, not related to performance, are protection and priority. The above discussion suggests that the requirements specified by the initiating DLS user may not match those required by the responding user. Or, these may not be supported by a provider. In either case, connection is not established. This situation results in primitive sequences shown in Figure 4.11 (b) and (c). Refusal to accept or to support a connection is indicated by issuing a DL-DIS-connect primitive. Commenting upon other parameters of DL-CONNECT primitives, the Responding Address is usually the same as the Called Address. Further, some Data Link protocols may not permit User Data to be included in DL-CONNECT primitives.

Table 4.3 Data Link Services and Their Parameters.

Service	Primitive	Parameters
Connection Establishment	DL-CONNECT request DL-CONNECT indication DL-CONNECT response DL-CONNCET confirm	see Table 4.5 see Table 4.4 see Table 4.4 see Table 4.4
Connection Release	DL-DISCONNECT request DL-DISCONNECT indication	Reason Originator, Reason
Normal data Transfer	DL-DATA request DL-DATA indication	user data user data
Expedited data Transfer	DL-EXPEDITED-DATA request DL-EXPEDITED-DATA indication	user data user data
Connection Reset	DL-RESET request DL-RESET indication DL-RESET response DL-RESET confirm	Reason Originator, Reason
Error reporting	DL-ERROR-REPORT indication	Reason
Data transfer (connection-less)	DL-UNIT-DATA request DL-UNIT-DATA indication	see Table 4.5 see Table 4.5

Parameter	DL-Connect Request	DL-Connect indication	DL-Connect response	DL-Connect confirm
Called Address	X	X		
Calling Address	X	X		
Responding Address			X	X
Expedited Data Selection	X	X	X	X (=)
Quality of Service	X	X	X	X (=)
DLS User Data	X	X (=)	X	X (=)

Note: (=): The parameter value is equal to its value in preceding primitive.

Table 4.4 Parameters of the DL-Connect Primitives.

2. Connection release: DL-DISCONNECT service is both unconfirmed as well as provider-initiated. Further, when a connection is released, all data in transit is discarded. Preferably, users must ensure that there is no data is transit before issuing a DL-DISCONNECT request, unless a user or the provider is forced to do so due to errors. The parameters Originator and Reason obviously specify whether the release is initiated by a user or the provider, and the reason, if known. The value of the parameter Originator is one of DLS User, DLS Provider, or Unknown. The value of the parameter Reason depends upon the value of the Originator. Example values of Reason are Connection rejection, DLSAP unreachable, permanent condition, Disconnection, abnormal condition, and Reason unspecified. Figure 4.12(a), (b) and (c) illustrate the release of a connection by a DLS user, the DLS provider, or by both simultaneously.

3. Normal data transfer: Normal data transfer is an unconfirmed service (see Figure 4.12(d)). But, within the layer, Data Link entities may acknowledge the receipt of Data Link Protocol Data Units that contain user data. In other words, a user can be assured that its DLSDU will be delivered without errors, unless the connection is reset or released.

4. Expedited data transfer: When available, the use of the service is negotiated during connection establishment. Expedited DLSDUs are sent on an urgent basis, and are subject to flow control different from that applicable to normal user data. (See Figure 4.12(e) for an illustration of the use of DL-EXPEDITED DATA primitives.)

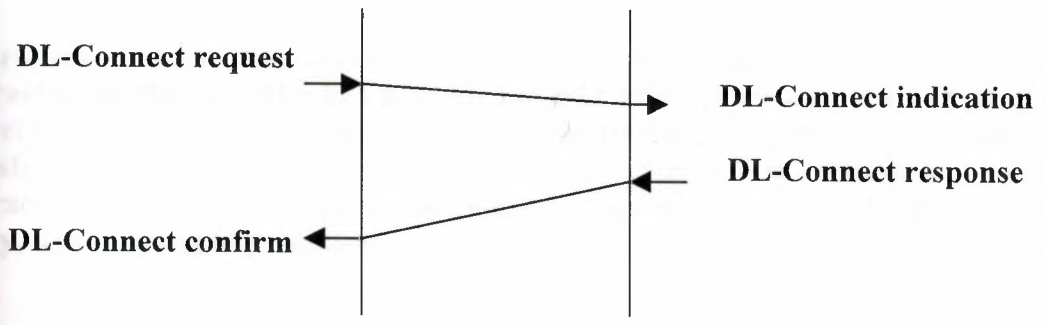
5. Connection reset: The service is a confirmed service. A reset may also be initiated by the DLS provider. The net effect is to discard all data in transit. The parameters Originator and Reason identify the source and the reason for doing so. If the value of the Originator parameter is DLS Provider, then the value of the Reason parameter is one of Data Link flow control congestion or Data Link error. If Originator is DLS User, then the value of Reason parameter is User Synchronisation. Figure 4.12(f) and (g) illustrates the sequences of primitives for situations where the two DLS users, or a user and the DLS provider, issue a DL-RESET primitive at about the same time.

Table 4.5 Parameters of the DL-Unit data Primitives

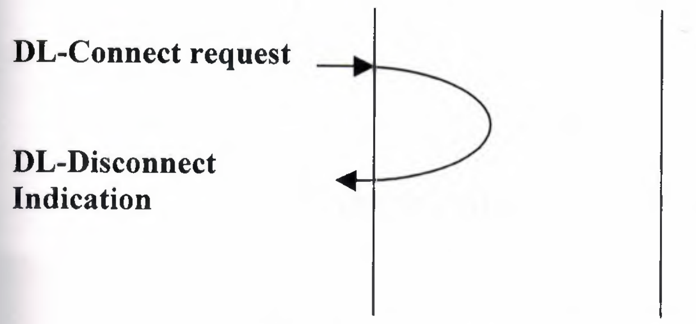
Parameter	DL-Unit data request	DL-Unit data indication
Called Address	X	X(=)
Calling Address	X	X(=)
Quality of service	X	
DLS User Data	X	X(=)

Note: (=): The parameter value is equal to its value in preceding primitive.

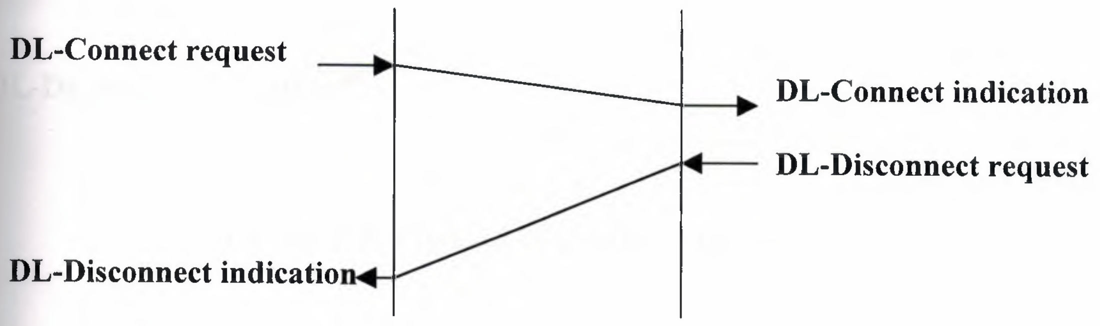
6. Error reporting: The optional Error Report service is used by the service provider to inform a user that an error has occurred. This error may result in the loss of one or more DLSDUs. Currently, most Data Link protocols simply reset the connection, instead of reporting the error and proceeding. This service is provider-initiated, and is illustrated in Figure 4.12(h).



(a) Successful connection establishment.



(b) Connection rejected by the DLS provider.

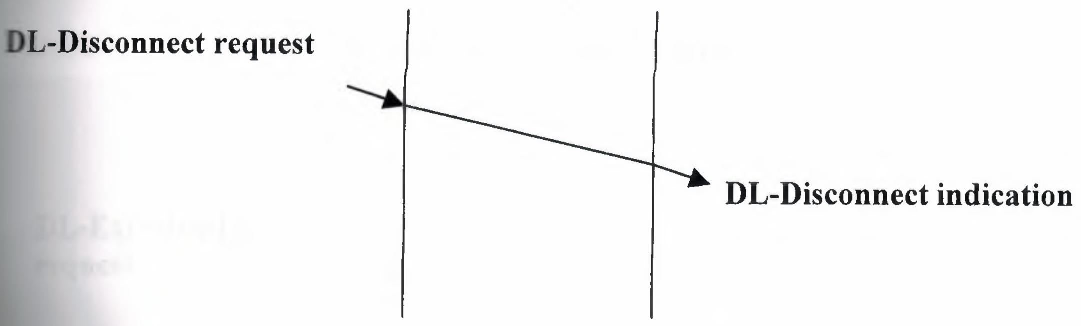


(c) Connection rejected by the other DLS provider.

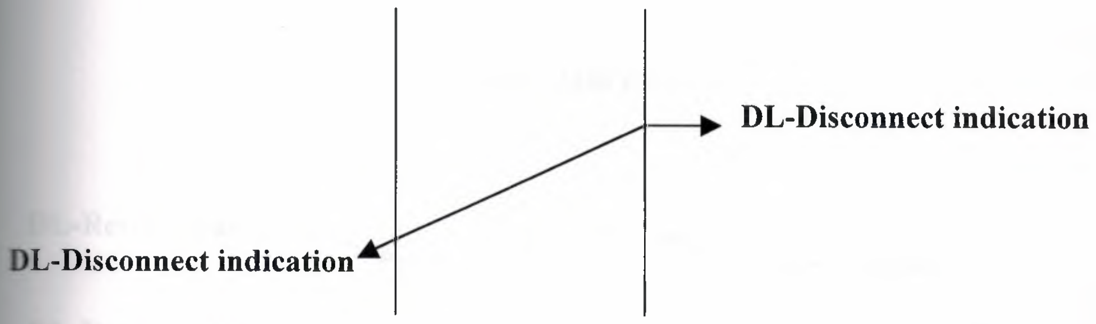
Figure 4.11 Example use of DL-Connect primitives.

6. Connection-less data transfer: In local area networks, particularly, the Data Link may simply provide connection-less data transfer. A DLSDU is made available to the service provider together with a Calling and a Called Address. The quality of service expected of the Data Link layer also specified. Again, this service is unconfirmed. Within the layer, Data Link entities may not acknowledge the receipt of a protocol data unit containing a DLSDU. Figure 4.12(i) illustrates the use of corresponding primitives. Figures 4.11 and 4.12 give some idea of how the use of Data Link service involves issuing

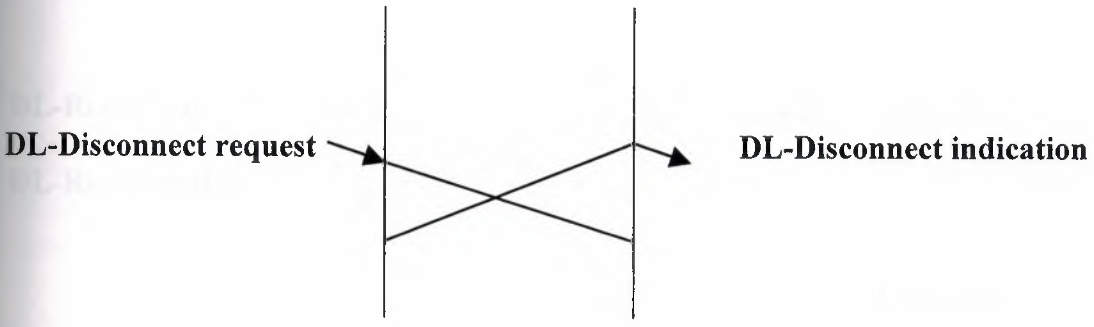
of primitives at the two service access points. A formal model which specifies the interaction at the two DLSAPs is given in [ISO 8886.2] (see also [CCITT X.212]). The model is based on two queues between the two DLSAPs, one for each direction of data transfer. It specifies in an abstract manner the cause-effect relationship between the interactions at the two service access points. It does not, however, specify the constrains, if any, upon issuing of primitives by a user or the service provider at a given DLSAP.



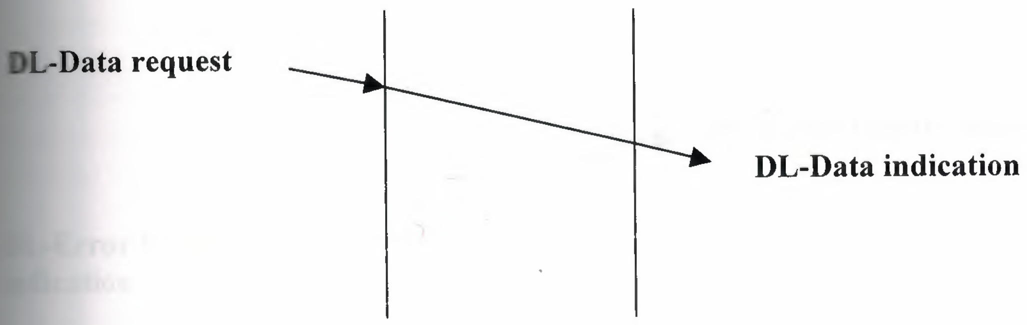
(a) Connection release by a DLS user.



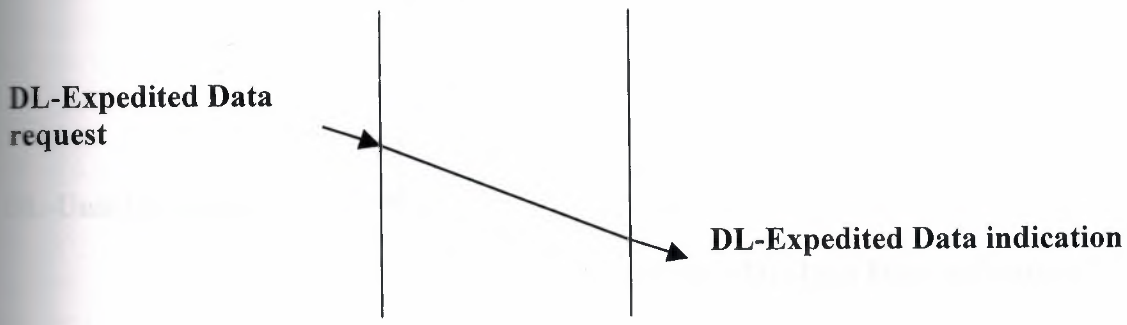
(b) Connection release by the DLS provider.



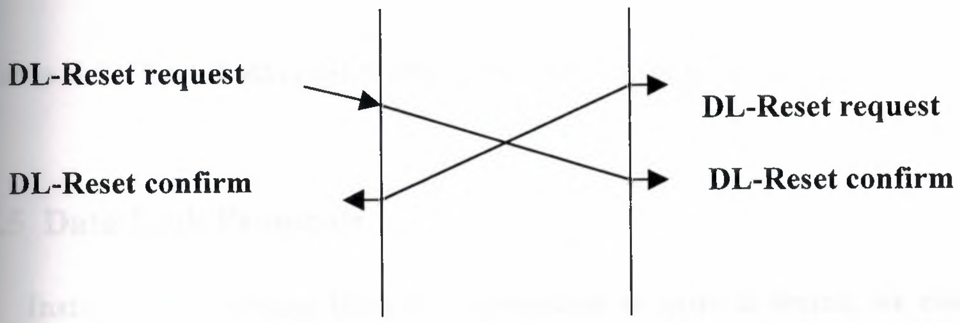
(c) Connection release simultaneously by a user and a provider.



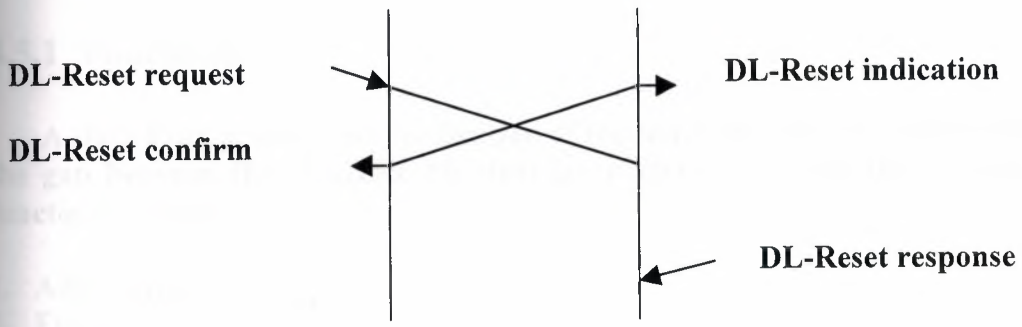
(d) Normal transfer data.



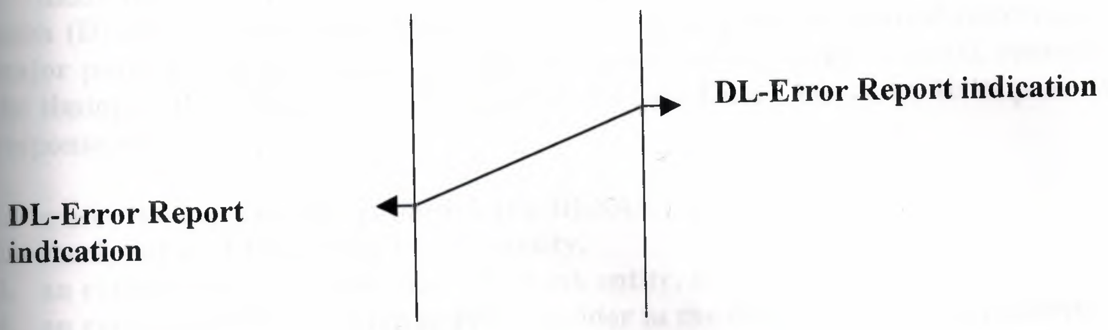
(e) Expedited data transfer.



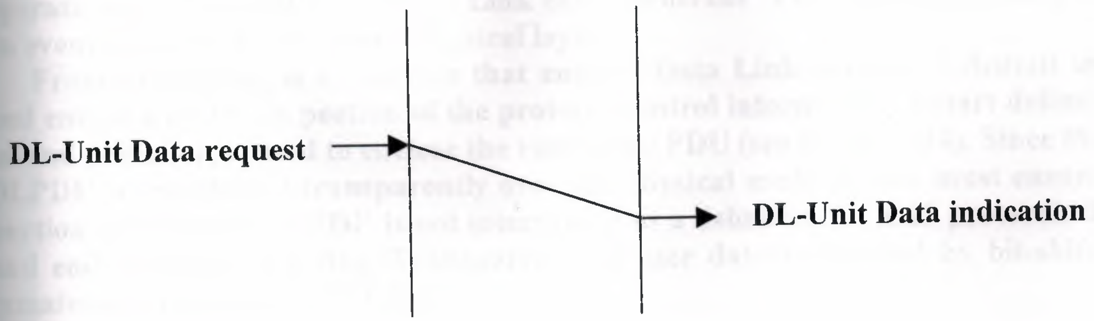
(f) Connection reset by the two users simultaneously.



(g) Connection reset by a DLS user and the DLS provider simultaneously.



(h) Error reporting.



(i) Connection-less data transfer.

Table 4.12 Use of other DLS primitives: some examples.

4.5 Data Link Protocols

Instead of discussing Data Link protocols in general terms, we concentrate on two Data Link protocols, CCITT's X.25 LAPB and IEEE's 802.2 and 802.3. The protocols broadly cover both wide area and local area networks.

4.5.1 Functions

A Data Link protocol is specification of the functions that are implemented to bridge the gap between the available Physical layer service and the Data Link service. The functions include:

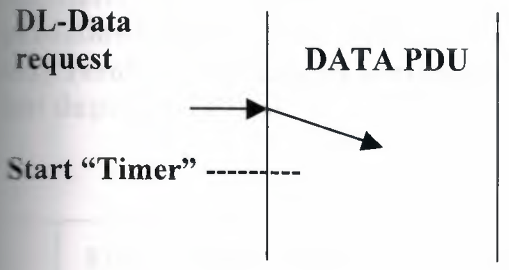
1. Addressing,
2. Frame delimiting,
3. Error detection, recovery and sequencing,
4. Flow control,
5. Protocol error detection and notification.

Aside from processing information, Data Link entities exchange DL protocol data units (DLPDU), which contain user data as well as protocol control information. The major portion of a protocol specification relates to the syntax (format), semantics and the timing of DLPDUs. A DLPDU may be sent by a Data Link entity to its peer entity in response to:

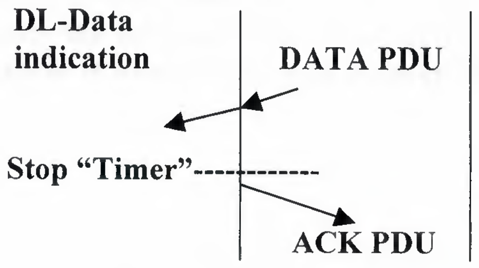
1. a user issuing a service primitive at a DLSAP,
2. receipt of a DLPDU from its peer entity,
3. an event occurring within the Data Link entity, or
4. an event notified by the lower PhS provider in the form of a service primitive.

When one of such events occurs, the Data Link entity may simply issue a service primitive either at the corresponding DLSAP or PhSAP, or initiate some activity which is local to the entity itself. Figure 4.13 illustrates these cases. Note that "Timer" and its operations are internal to the Data Link entity, whereas "Physical connection failure" is an event signalled by the lower Physical layer.

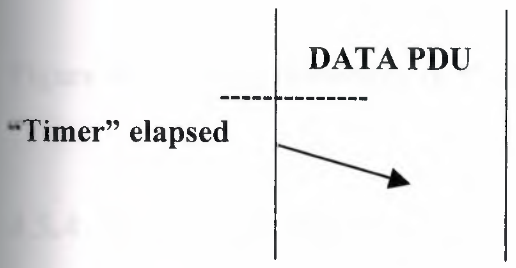
Frame-delimiting is a function that enables Data Link entities to delimit the start and end of a DLPU. A portion of the protocol control information, a start delimiter and an end delimiter, is used to enclose the remaining PDU (see Figure 4.14). Since the entire DLPDU is transferred transparently over the Physical medium, one must ensure that a portion of the enclosed PDU is not interpreted as a delimiter. In X.25 protocol, the start and end delimiter is a flag. Transparency of user data is ensured by bit-shifting the remaining PDU (see [CCITT X.25]).



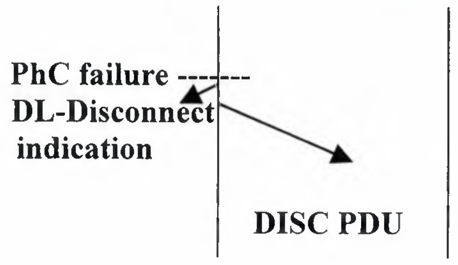
(a) User issues a primitive.



(b) Receipt of a DL PDU.



(c) Event occurs within the DL-entity.



(d) Event notified by the Physical layer.

Figure 4.13 Occurrence of incoming events and resulting actions: an illustration.

4.5.3 Error Detection, Recovery and Sequencing

Error detection, recovery and sequencing functions are all concerned with ensuring that DLSDUs are delivered to the receiver entity without error. Recall that transfer of a sequence of bits over the physical medium is prone to transmission errors. Further, since processing capability and buffer space is always finite, a receiver may not be able to buffer incoming DLPDUs. In spite of these limitations, DLSDUs must be delivered without alteration in their contents, loss, or duplication, and in the proper sequence.

There are a number of approaches to error detection and recovery, but the one most frequently employed is based upon error detection and re-transmission. If a PDU is known to have been corrupted by noise it is simply discarded. A discarded PDU can now be treated in a manner identical to that when it is lost. Transmission errors can be detected by computing a checksum on the PDU and transmitting it as part of the PDU itself. Thus, a checksum is considered to be a part of the protocol control information. The choice of checksum algorithm is based upon the characteristics of transmission error. For example, a Cyclic Redundancy Checksum (or CRC) is considered ideally suited for detecting burst errors. Further, the generation of a CRC, or its interpretation, can be conveniently done in hardware.

Lost PDUs can be recovered using one of several protocols, based on re-transmissions. Once a PDU is known to have been lost, it is simply re-transmitted. Thus detection of loss of PDUs becomes a major concern. In most protocols, the responsibility of ensuring that a PDU has been delivered to the corresponding Data Link entity rests with the sender. The receiving Data Link entity is expected to acknowledge the receipt by sending an appropriate acknowledgement. Acknowledgement PDUs are themselves not acknowledged. Thus, if an acknowledgement does not arrive within a predefined interval, the initiator simply re-transmits the original PDU. Of course, this may result in duplication sometimes. Thus, a protocol is needed that can recover from lost duplicated PDUs.

	Flag	Other header information and user data	Flag	Flag	

Note: Flag = '01111110'

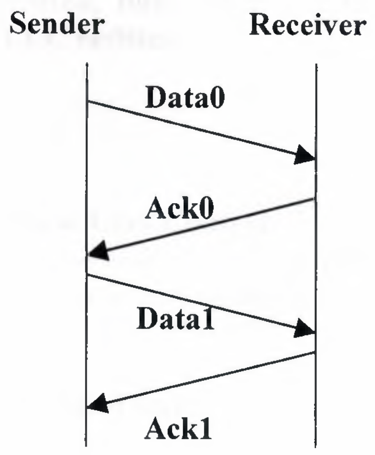
Figure 4.14 Frame delimiting in X.25 Data Link protocol.

4.5.4 Alternating-Bit Protocol

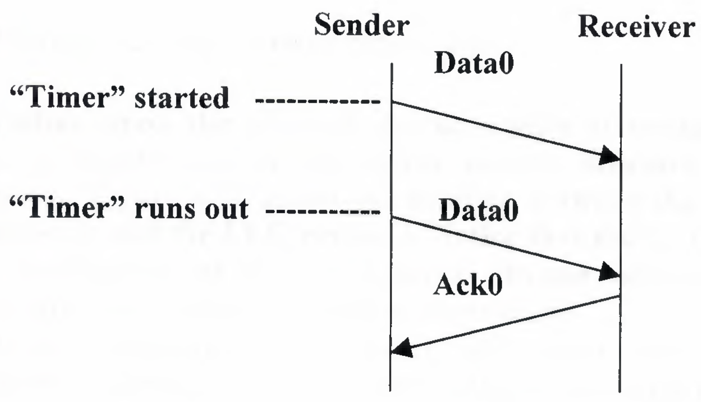
Elsewhere in the literature the protocol is also called stop-and-wait protocol. For simplicity, we shall consider reliable transfer of PDUs that carry user data, but only in one direction. Such a PDU shall be referred to as a Data PDU. A PDU sent as acknowledgement is called Ack PDU. Further, the Data Link entity which sends Data PDUs is called the sender, whereas the entity which receives Data PDUs, and sends Ack PDUs, will be termed the receiver. The Physical layer is assumed to support-full-duplex transfer of PDUs. PDUs are received in the proper sequence, but may be lost. No upper bound is assumed on the delay n transferring a PDU.

Each Data PDU is sequentially numbered modulo-2. Similarly, an Ack PDU carries the sequence number, 0 or 1, of the Data PDU being acknowledged. The sender maintains a timer, which is started soon after a Data PDU is sent. It is stopped as soon as an outstanding Data PDU is acknowledged by the receiver. When the timer runs out, the Data PDU is re-transmitted, and the timer restarted. Sample PDU transmissions are shown in Figure 4.16, covering four different situations,

1. error-free and timely transfer of Data and Ack PDUs,
2. the Data PDU is lost,
3. the Ack PDU is lost,
4. delay in acknowledging the Data PDU.



(a) PDUs are transferred error-free and without delay.



(b) Data PDU is lost.

Figure 4.15 The alternating-bit protocol: an illustration.

4.6 Local Area Networks

The Data Link layer in local area networks is sub-divided into two sub-layers, called Media Access Control layer (or MAC) and Logical Link Control layer (or LLC). This is illustrated in Figure 4.16. Each Data Link entity has equal access to this medium, and all transmissions are broadcast over the medium. However, only the addressed receiver(s) may buffer incoming PDU. The mechanism used to co-ordinate access to the medium by all contending systems on the network is called media access control scheme. Other functions implemented within the MAC sub-layer include frame-delimiting, addressing, and error detection.

Thus, the MAC layer may be viewed as a sub-layer that provides a service using which LLC entities may transfer a protocol data unit without being concerned with how to access the broadcast medium, or with frame-delimiting, addressing, and error detection. The Logical Link Control layer uses this service the provide a connection-less or both vice is connection-oriented, flow control, error recovery and re-sequencing functions are implemented by LLC entities.

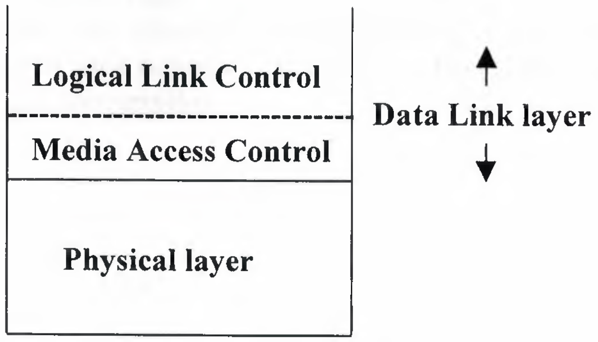


Figure 4.16 Sub-layers in local area networks.

4.6.1 Media Access Control Sub-Layer

Depending upon the physical characteristics of medium and the topology of the network, a number of media access control schemes have been developed and standardised. Figure 4.17 shows the relation between the three commonly used MAC layer protocols and the LLC protocol. Notice that the LLC protocol can use any MAC layer is specification of the media access scheme together with that of physical and electrical interface with the transmission medium.

Table 4.6 summarises the major differences between the three MAC layer specifications. Below we consider the contexts in which each protocol is particularly suited. A network based upon CSMA/CD scheme over a 10 Mbps co-axial cable is particularly suited in those environments where the volume of traffic is low and minimum channel access delay is desirable. There is, however, no upper bound on the access delay, which implies that it may not be suited for real time applications, specially if the volume of traffic is high.

Applications that require a high throughput, and are not as particular about access delay, may benefit from the use of a 4 Mbps ring that uses Token-passing channel access scheme. An upper bound on the delay can be computed and enforced.

Table 4.6 A Summary of the Specification of IEEE's MAC and Physical Layer Standards.

Characteristics	IEEE 802.3	IEEE 802.4	IEEE 802.5
Channel Access	CSMA/CD bus	Token-passing bus	Token ring
Data rate	10 Mbps	1, 5, 10 Mbps	1,4 Mbps
Trunk cable	50 ohm co-axial	75 ohm co-axial	twisted pair
Topology	omnidirectional bus	omnidirectional/ directional bus	ring
Base-band/broadband	base-band	both	base-band
Bit-level encoding	Manchester	Manchester modulation	differential Manchester

The IEEE 802.4 standard has provision for a variety of transmission rates and media. The most interesting, perhaps, is its implementation over a broadband CATV cable based bus. It offers multiple high speed channels, some of which may be used to carry voice or video signals. The channel access scheme is token-passing which, as before, offers high throughput and bounded delay. It is, therefore, suited for real time applications, including factory automation.

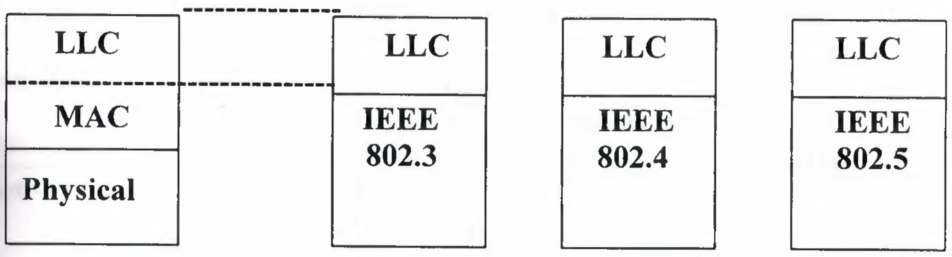


Figure 4.17 IEEE 802 standards and their relation to OSI layers.

4.6.2 Logical Link Control Sub-Layer Services

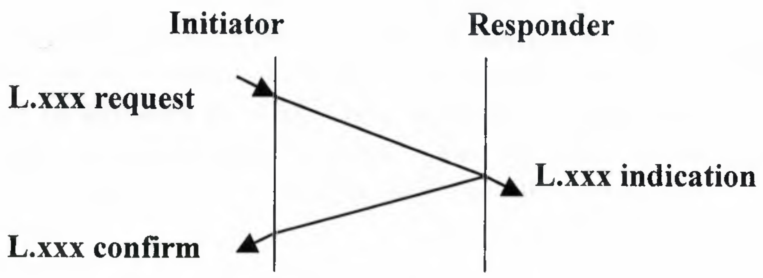
The LLC sub-layer offers to Network entities two classes of service. Class 1 service is connection-less, and is relevant to those applications that do not require error-free or flow-controlled transfer of user data. Class 2 service is both connection-oriented and connection-less. The standard IEEE 802.2, however, defines these classes in terms of types of operations. Type 1 operation is simply connection-less data transfer, whereas Type 2 operation a balanced-mode connection is required to be established before data transfer can take place.

A summary of LLC service primitives and their parameters is given in Table 4.7. First, note that there is no response primitive at all. That is, any confirmed service, connection establishment or reset for instance, the responding Network entity does not issue a primitive in response to an indication primitive. Of course, the responding LLC service user is simply informed of the operation. But, a confirm primitive issued to the initiating LLC service user is based on the acknowledgement received from the remote LLC entity. Figure 4.18 illustrates the sequence of primitives that are issued in order to establish, reset or disconnect a connection, or to transfer data. Note that connection establishment, reset, disconnection and data transfer are confirmed services. It may be pointed out that the Status information provided in a confirm primitive has remote significance.

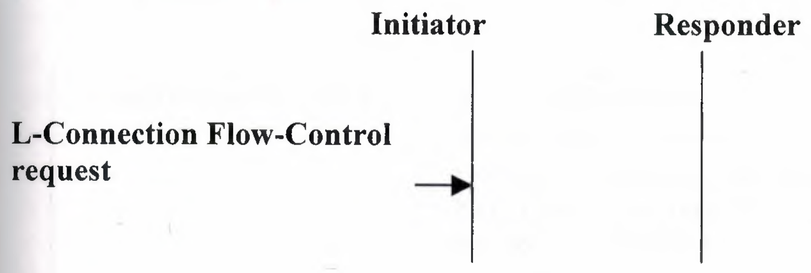
The other major difference is that expedited data transfer and error reporting are not supported. Instead, user may authorise the LLC layer to receive only a specified amount of data. There is, however, no correspondence between the Flow-Control request and Flow-Control indication primitives. These are independent service primitives, and have local significance only. Lastly, connection-less data transfer is unconfirmed, but a confirm primitive, with local significance, is issued.

Table 4.7 LLC Service Primitives and Their Parameters

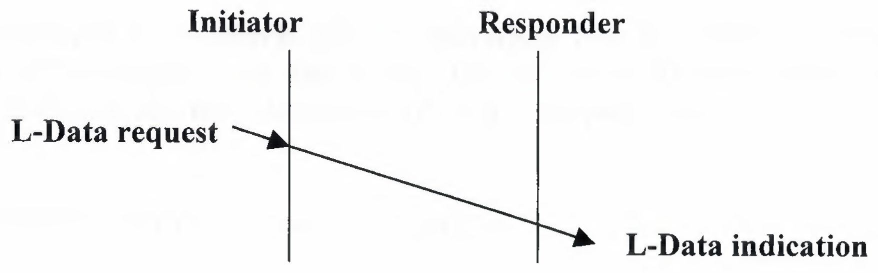
Service	Primitives	Parameters
Connection Establishment	L-Connect request L-Connect indication L-Connect confirm	service class service class, status service class, status
Data transfer	L-Data-Connect request L-Data-Connect indication L-Data-Connect confirm	user data user data status
Release	L-Disconnect request L-Disconnect indication L-Disconnect confirm	- reason status
Reset	L-Reset request L-Reset indication L-Reset confirm	- reason status
Flow control	L-Flow-Control request L-Flow-Control indication	amount of data amount of data
Connection-less Data Transfer	L-Data request L-Data indication	user data, service class user data, service class



(a) Connection establishment, reset, disconnection or connection-oriented data transfer.



(b) Flow control.



(c) Connection-less data transfer.

(Note: Above it is assumed that a service is not initiated simultaneously by two users or a user and the LLC layer.)

Figure 4.18 Sequence of LLC service primitives.

4.6.3 Logical Link Control Sub-Layer

The functions implemented in Class 2 service, and the corresponding procedures, are similar to those of X.25 Data Link layer. A major difference between the two in the modules used to sequentially number Data PDUs and, as a consequence, in the PDU formats. The Data PDUs are numbered modulo-128, which is basically a recognition of the fact that the protocol is for use over high speed networks where it is desirable to have as many outstanding Data PDUs as possible, subject to buffer availability.

An LLC PDU carries addresses of the source and destination LLC service access points. These are in addition to MAC layer addresses. But, as with X.25 protocol, only one data link may be established between a pair of LLC entities, unless each entity supports services at more than one service access point.

Table 4.8 Connection-Oriented and Connection-Less Network Service Elements

Connection-oriented	NC establishment Normal data transfer Receipt Confirmation (optional) Expedited data transfer (optional) NC Reset (optional) NC release
Connection-less	Unit data transfer

4.7 Network Services

A formal definition of Network objects, including NSAPs, Network connections, NCEPs, and NCEP identifiers was also given. The discussion in this section is mainly concerned with Network service primitives and with their parameters.

4.7.1 Connection-Oriented Service Elements

Table 4.8 summarises the service elements available to NS users. Connection-oriented Network service is a collection of service elements that allow its users: (a) to establish or release connections, (b) to transfer data transparently (either normally or an urgent basis), (c) to acknowledge the receipt of normal data, or (d) to re-initialise connection.

1. NC Establishment: An NS user may establish a Network connection (NC) with another NS user. The address of the NSAP to which the responding NS user is attached is assumed to be known to the initiating NS user. During NC establishment, an NS user may request, and negotiate with the other NS user and the NS provider, the quality of service to be provided over the NC.
2. NC Release: Either NS user may unilaterally and unconditionally release the NC once a connection has been established, or even during the establishment phase. As one consequence, any user data currently in transit may not be delivered, and discarded. Alternatively, a connection may be released by the NS provider, if it determines that it is no longer possible to support the connection, either due to breakdown or deterioration in the quality of service.
3. Normal Data Transfer: NS users may exchange data, in the form of Network Service Data Units (or NSDUs) consisting of an integral number of octets, such that the boundaries between NSDUs and their contents are preserved at the two ends. A receiving NS user may control the rate at which an NS user sends data.
4. Receipt Confirmation: By itself, when an NSDU is delivered to the destination Ns user, the NS provider does not confirm its delivery to the initiating NS user. If the users so desire, and if the Receipt Confirmation service is provided by the Network layer, an NS user may acknowledge the receipt of an NSDU.
5. Expedited Data Transfer: Transfer of a limited amount of user data on an urgent basis may be requested by an NS user. But, this service is available only when the NS users agree to use it and the NS provider agrees to provide it. Further, the transfer of

Expedited-NSDUs may be subject a similar, but distinct, flow control by a receiving NS user.

6. Reset: A reset, or a re-initialisation, of the established connection may be initiated by either NS user or by the service provider, provided the service is available and its use has been negotiated at the time of establishing the connection. The net effect is to restore the connection to a state where there is no data within the network. All data with Network entities or in transit is discarded. From the NS users viewpoint, the service may be used to resynchronise their states, in case they detect errors within the Transport layer.

4.7.2 Connection-less Service Element

Connection-less data transfer, on the other hand, does no require the establishment of a connection prior to data transfer. Thus, the only available service element relates to Connection-less Data Transfer. An NS user may transparently transfer an NSDU, of a fixed maximum length, to another NS user is attached, is provided by the sending NS user. Each NSDU is sent independent of other NSDUs together with the address of the source and destination NSAP. While initiating the transfer, the sending NS user may request a desired quality of service that the NS provider must associate with the transfer. The NS provider is expected to make every attempt to deliver the message, correctly and timely. There is no guarantee, however, that the data would be delivered correctly, or delivered at all.

4.7.3 Service Primitives and Parameters

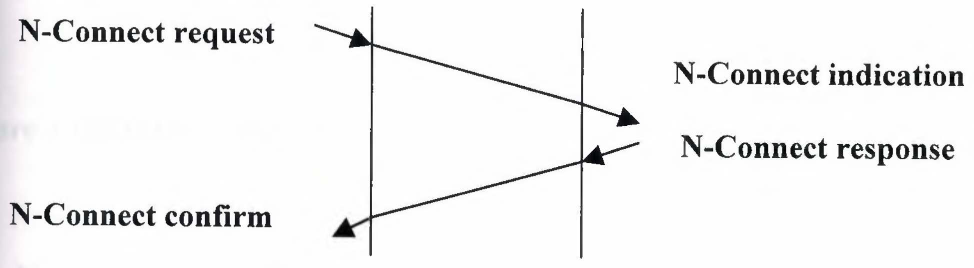
Note, that NC Release is unconfirmed as well as provider-initiated, whereas NC Reset is confirmed and provider-initiated normal data transfer is confirmed, but users may acknowledge receipt of data using the unconfirmed Receipt Confirmation service.

1. The Calling and the Called Addresses of the N-CONNECT primitives refer to the addresses of the NSAPs to which the initiating and the responding NS users are attached. More often than not, the Responding Address in the corresponding response and confirm primitive is identical to the Called Address. However, in case of re-direction or generic addressing the value of Responding Address may be the address of the NSAP to which the connection has been established or should be established by the Calling NS user entity.
2. N-COONECT parameters, Receipt Confirmation Selection and Expedited Data Selection parameters enable NS users and the NS provider the negotiate the availability, and use, of the corresponding optional service elements. The negotiation procedure, for each selection, is such that if either one of the users or the provider does not agree to its availability or its use then the service is not used.
3. While a number of quality of service parameters have been defined, only Throughput and Transit Delay are negotiated.
4. The parameters, originator and reason, may be used to convey the source (NS user or NS provider) of disconnection and the reason, if known. The parameter, Responding Address, relevant only when a connection request is refused by the corresponding user, conveys the address may be different from the Called Address, in case of re-direction or generic addressing.

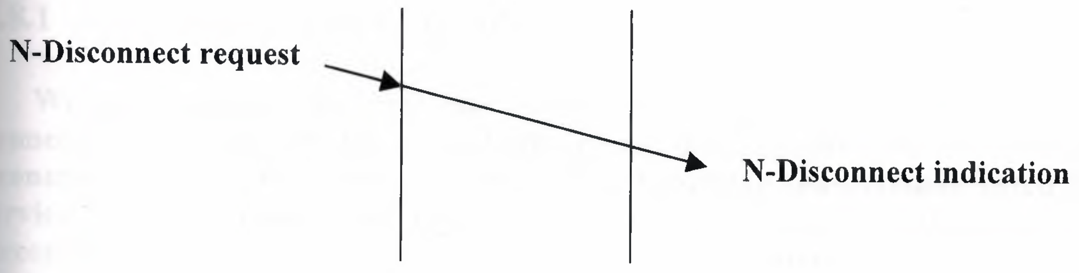
5. User data in N-CONNECT primitives is optional. Further, while it is specified as one of the parameters in N-DISCONNECT primitives, it may not be available in some networks. When an implementation supports transfer of user data in N-CONNECT or N-DISCONNECT primitives, its length is limited.
6. Once a Network connection has been established, at each end the NS user and the NS provider refer to its using an NCEP Identifier. This identifier is assigned by the NS provider at the time of connection establishment and make known to the local NS user. Since it has local significance, the identifier does not appear as a formal parameter of N-CONNECT primitives.

4.7.4 A Queue Model of Network Service

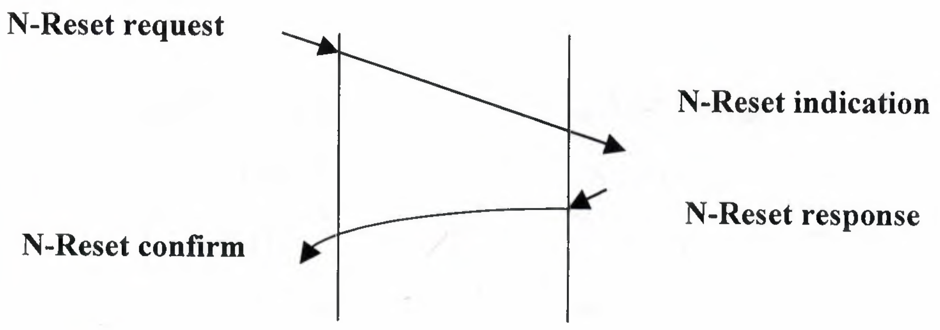
From Figure 4.19, when a primitive is issued at an NSAP, for a specific connection, a corresponding primitives is subsequently issued by the NS provider at the other end of the connection. A model is, therefore, required to specify such a correspondence between interactions at two NCEPs. Such a model, based on queues. The application of that model to Network service is described here. This model is only an abstraction, and may only be used to guide an implementation of the Network layer.



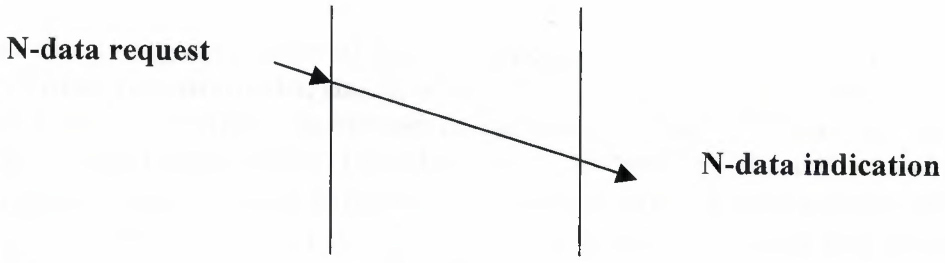
(a) Connection establishment.



(b) Disconnection.



(c) Reset.



(d) Normal data transfer.

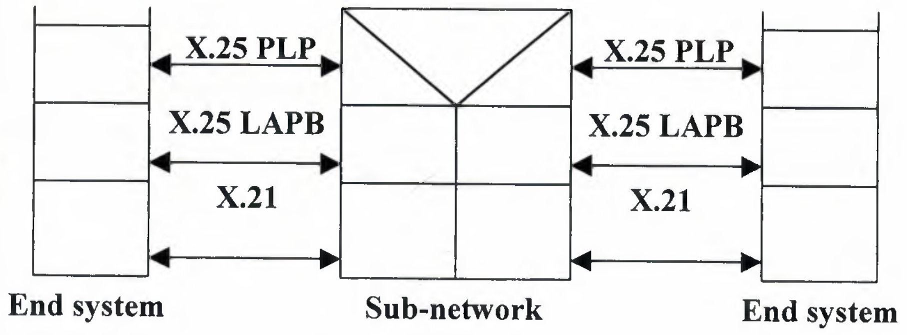
Figure 4.19 Typical sequences of primitives.

4.8 Network Layer Protocols

We cover both area and local area networks, but limit ourselves to a network consisting of one sub-network only.

4.8.1 X.25 Packet-Level Protocol

We have already seen that the Network service may be connection-oriented or connection-less. Further, the Data Link service may also be connection-oriented or connection-less. We first discuss a protocol for providing connection-oriented Network service using a connection-oriented Data Link service. It based on CCITT's Recommendation X.25 (see Figure 4.20) and is a specification of a sub-network access protocol that allows Network entities in end systems (also called packet-mode DTEs) to interface with a packet-switched sub-network. the access protocol at the Network layer is connection-oriented. That is, using X.25 protocol, Network entities in end systems can establish connections between themselves. Each connection is end-to-end, although the exchange of protocol data units is only between an end system entity (or DTE) and the sub-network to maintain correspondence between the two segments of the connection, and thereby, relay the semantics of each PDU across the sub-network.

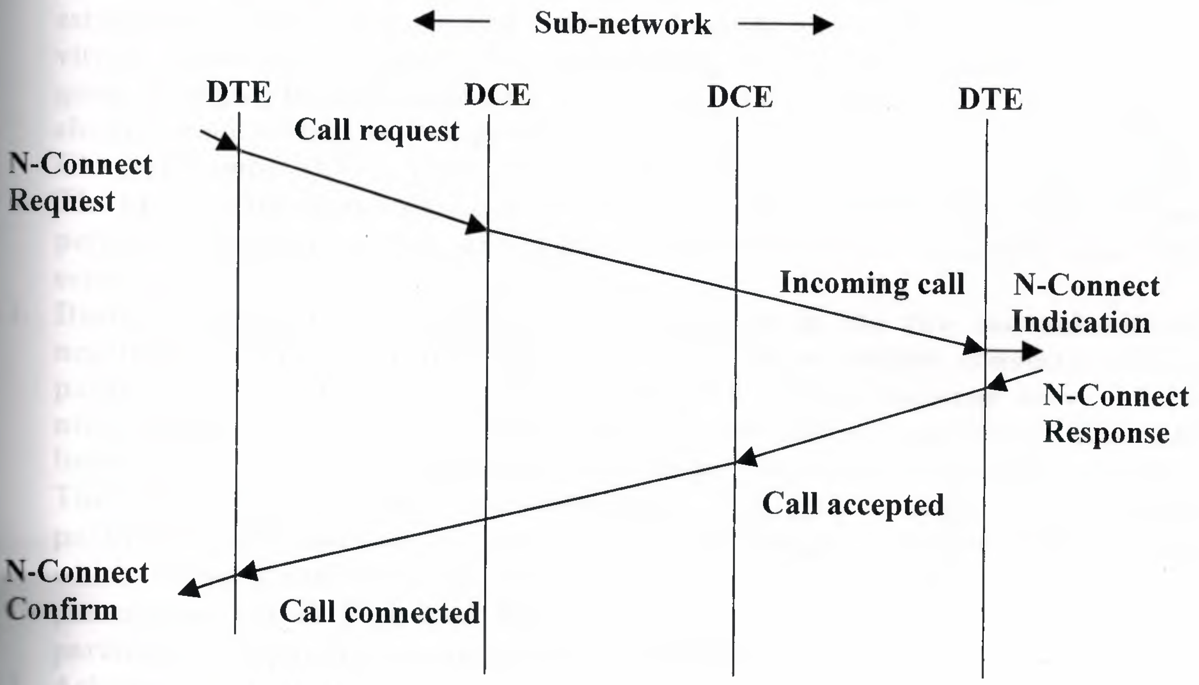


Note: PLP = Packet – level protocol.

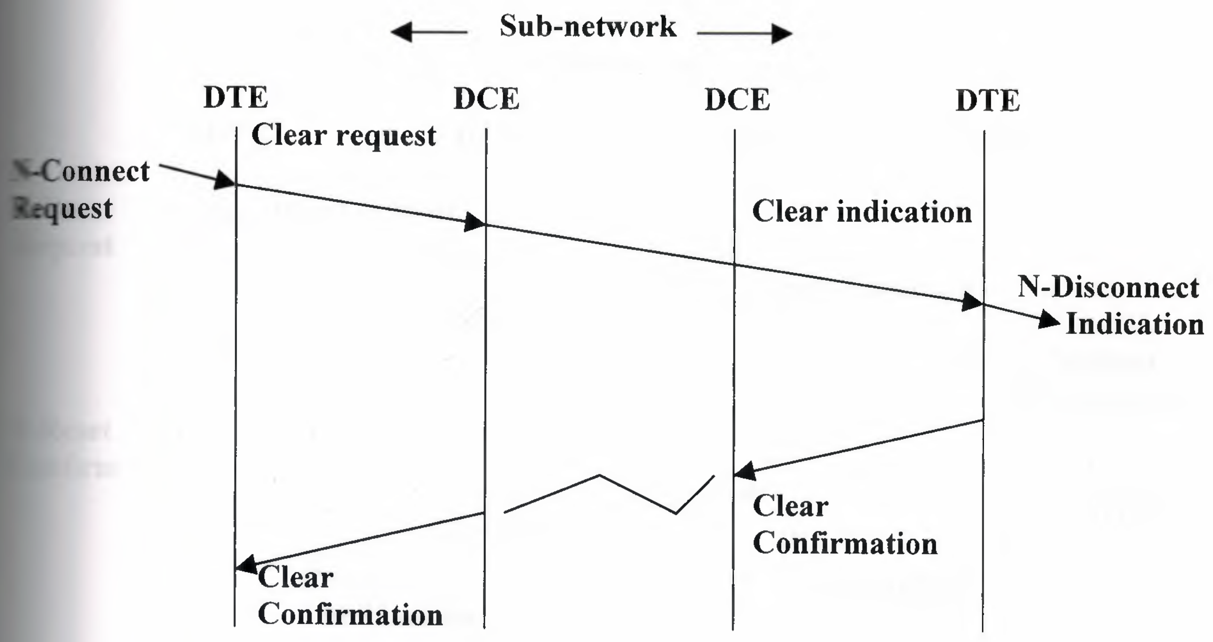
Figure 4.20 Sub-network access protocol: X.25.

Before discussing the details of the X.25 protocol, it is important to point out that in the case of wide area networks, the Network layer X.25 protocol assumes the availability of a Data Link connection established in conformity with X.25 link access procedure (balanced or unbalanced LAP). Therefore, all X.25 Network layer Protocol Data Units are transferred as user data in Information frames of the link access procedure.

Figure 4.21 illustrates the procedure for establishment, release and re-initialisation of connections, and for data transfer. Note that these figures do relate sending or receiving of PDUs to issuing of service primitives, although such a specification is not part of X.25. This relation can only be established it is clear that X.25 protocol can be used to provide connection-oriented Network service. Discussion of packet parameters is also postponed to next sub-section.



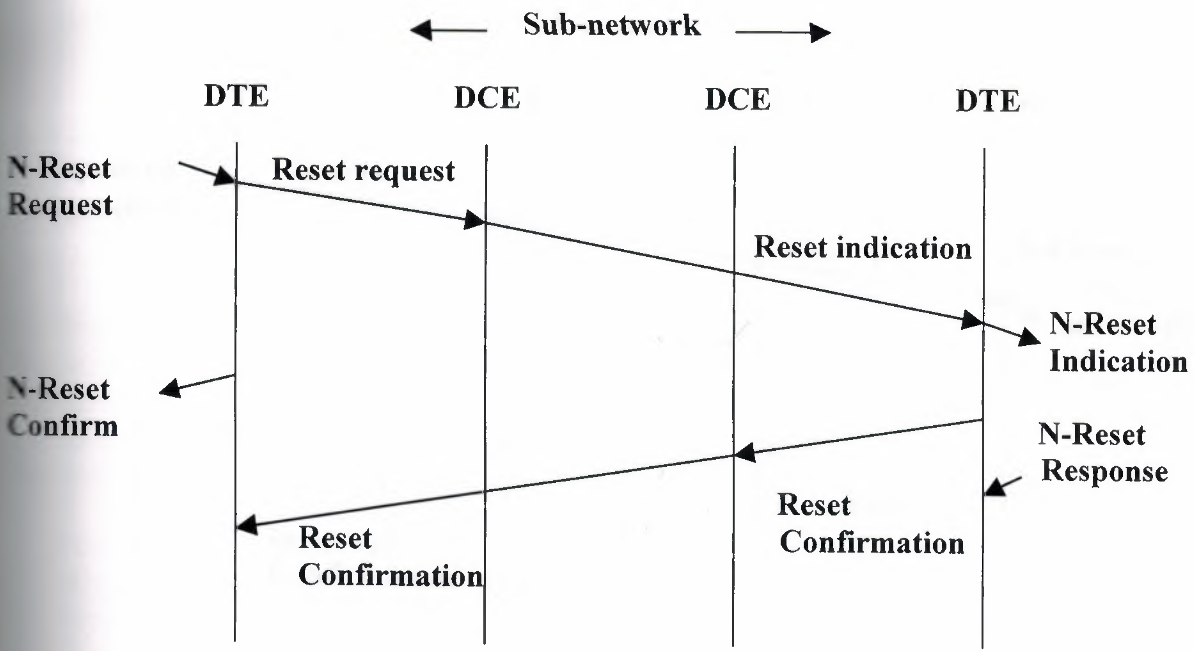
(a) Connection establishment.



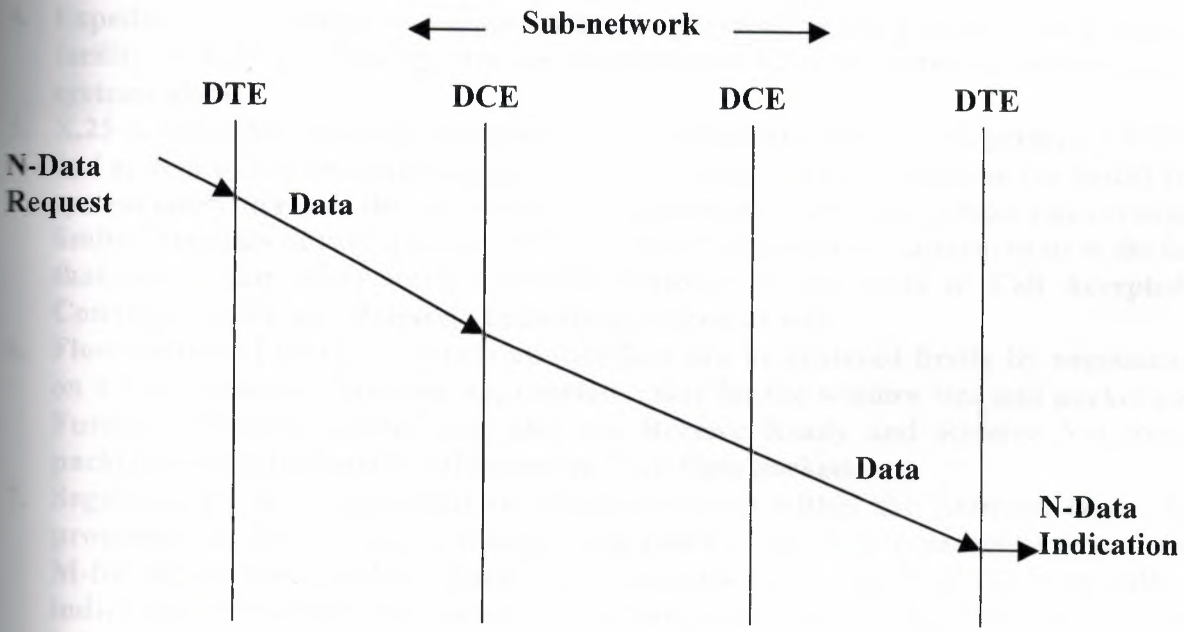
(b) Connection release.

Figure 4.21 X.25 protocol procedures.

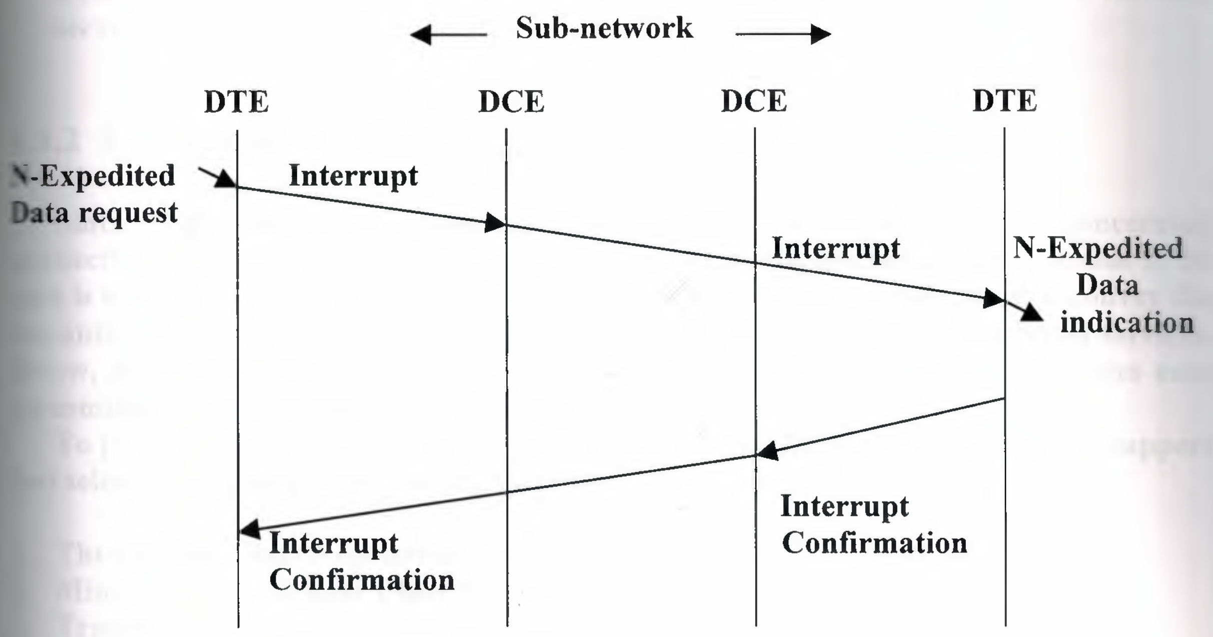
1. X.25 protocol provides a means to simultaneously maintain a number of connections, called virtual calls or permanent virtual circuits. Each virtual call goes through a call establishment, data transfer and clearing phase. It may also be reset. Permanent virtual circuits, on the other hand, are established on a permanent basis without going through a formal establishment procedure. These may only be reset, but never cleared. Each virtual call or permanent virtual circuits is identified by a Logical Channel Number (LCN), which serves the purpose of connection protocol identifier. The LCN is carried as a parameter in each X.25 PDU. Further, all virtual calls and permanent virtual circuits are possibly multiplexed onto a single Data Link connection.
2. During connection establishment Network entities in the two end sub-network negotiate, on Per connection basis, the values of a number connection-related parameters, and the use and availability of certain optional services. The negotiations can broadly be divided into two categories. Negotiations take place between Network entities and have significance for sub-network entities as well. These include end-to-end acknowledgement, reverse charging, flow control parameters, and fast select facility. The second category includes facilities using which Network entities in end systems negotiate, or simply convey, the value of parameters. Use of Expedited data transfer is one such example. Most of these parameters are passed as optional facility parameters.
3. Acknowledgements in X.25 may have an end-to-end significance (see Figure 4.21 (d)) or, the significance of an acknowledgement may be local to the host-sub-network interface. The D-bit is used to negotiate this facility during connection establishment, or to request an end-to-end acknowledgement.



(c) Connection reset.



(d) Data transfer.



(e) Expedited data transfer.

Figure 4.21 continued.

4. Expedited data transfer is supported by the protocol, and is known as an Interrupt facility in X.25 terminology. Its use is negotiated between Network entities in end systems alone.
5. X.25 protocol has the added provision for a fast-select facility. This permits a calling end system to request establishment of a connection with an option to the called end system entity to reject the connection. But, in doing so, the end systems can exchange limited amounts of user data in both directions. Of particular interest to us is the fact that use of fast select service permits inclusion of user data in Call Accepted / Connected and Clear Request / Indication packets as well.
6. Flow control of user data across an interface can be achieved firstly by negotiating, on a Per connection basis, an appropriate value for the window size and packet size. Further, Network entities may also use Receive Ready and Receive Not Ready packets to limit the number of incoming User-Data packets.
7. Segmentation is an important function performed within the Network layer. The procedure the perform segmentation is specified by the X.25 protocol in terms of an M-bit of User Data packets. Each packet contains one segment of user data with an indication of whether the packet is the last packet in the sequence, or not. This information is carried in the M-bit of User Data packets.
8. The Restart procedure in X.25 protocol may be used by a Network entity in an end system or the sub-network to initialise or re-initialise an interface. The consequence of restarting an interface is to clear all existing virtual calls, and to reset all permanent virtual circuits across the interface. This procedure is invariably used soon after a data link connection has been established or re-initialised.
9. Finally, there are some differences between the 1980, 1984 and the purposed 1988 versions of CCITT's Recommendation X.25. These differences may be significant

from the viewpoint of using the protocol the provide connection-oriented Network service.

4.8.2 Connection-Oriented Network Service using X.25 Protocol

Since X.25 protocol is basically connection-oriented, all aspects concerning connection management and data transfer over it are supported. What remains to be seen is whether X.25 protocol procedures and packet formats are adequate to convey the semantics of Network service primitives and to support negotiation of optional services. Below, we discuss these issues as also the mapping of Network service primitives onto transmission and reception of X.25 packets.

To provide connection-oriented Network service, the X.25 sub-network must support fast select service and the following facilities:

1. Throughput Class Negotiation,
2. Minimum Throughput Class Negotiation,
3. Transit Delay Selection and Indication ,
4. End-to-End Transit Delay Negotiation,
5. Calling Address Extension,
6. Called Address Extension,
7. Expedited Data Negotiation.

Table 4.9 Mapping of the Parameters of N-Connect Primitives

Parameters	Field or Facility
Called Address	Called DTE Address field Called Address Extension facility
Calling Address	Calling DTE Address field Calling Address Extension facility
Responding Address	Called DTE Address field Called Address Extension facility
Receipt Confirmation Selection	General Format Identifier
Expedited Data Selection	Expedited Data Negotiation facility
QOS Parameter Set	Throughput Class Negotiation facility Minimum Throughput Class Negotiation facility Transit Delay Selection and Indication facility End-to-End Transit Delay Negotiation facility
NS User Data	Calling or Called User Data field (Fast Select facility)

Figure 4.21 and 4.22 illustrate the correspondence between issuing service primitives and exchange packets between Network entities in end systems and the sub-network. table 4.9 summarises the mapping of parameters of N-CONNECT primitives onto different fields of the corresponding packets. Note that:

1. maintaining the correspondence between a Network connection and a virtual call is a local issue. What is significant is that a logical channel number enables the pair of Network entities at an interface to uniquely identify a virtual call.

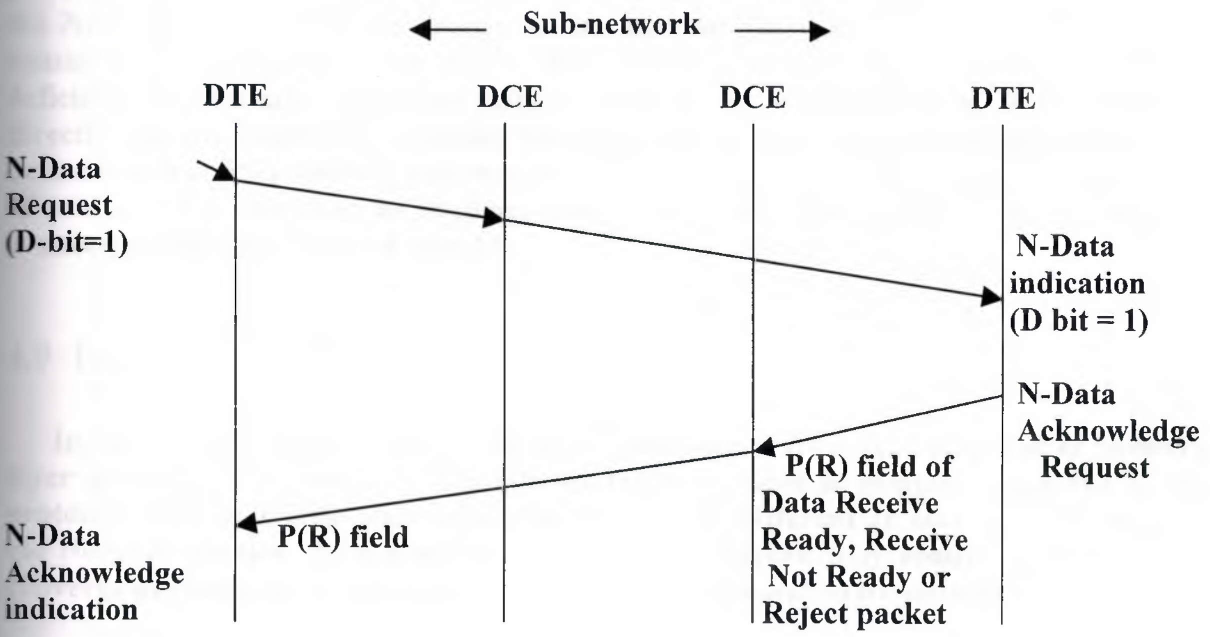

Sub-network
DTE
DCE
DCE
DTE
N-Data Request (D-bit=1)
N-Data indication (D bit = 1)
N-Data Acknowledge Request
P(R) field of Data Receive Ready, Receive Not Ready or Reject packet
P(R) field
N-Data Acknowledge indication

Figure 4.22 Receipt Confirmation.

2. The service parameters Calling and Called NSAP addresses are encoded as Calling and Called DTE Addresses provided the NSAP addresses can be deduced from the DTE addresses. This is the case when the Domain Specific Part of the NSAP addresses is absent, or equivalently, when only one NSAP is served by the Sub-network Point of Attachment (SNPA). In that case, NSAP address is the same as its DTE address. Otherwise, two or more NSAPs are served by an SNPA, the NSAP addresses are encoded using Calling and Called DTE Address Extension facilities, together with Calling and Called DTE Addresses. Further, since we are currently concerned only with a single sub-network, the DTE addresses are directly obtainable from the Initial Domain Identifier portion of the NSAP Address.
3. There is no X.25packet which specifically conveys the semantics of N-DATA ACKNOWLEDGE primitives. A variety of X.25 packets carry acknowledgement information in the form of a P(R) value. When a sender entity receives P(R) = (x + 1), it is an acknowledgement to all User Data packets sequentially numbered up to and including x. Whether this acknowledgement is local or end-to-end depends

upon whether the sender had requested an end-to-end acknowledgement with the User Data packet numbered x. If it is an end-to-end acknowledgement, the Network entity issues an N-DATA ACKNOWLEDGE indication primitive. At the remote receiver end, the Network entity delays sending an acknowledgement to a packet numbered x, until the corresponding NS user issues an N-DATA ACKNOWLEDGE request primitive. Needless to say, if an NSDU is segmented into a number of User Data packets, then end-to-end acknowledgement is sent only after the NSDU has been completely delivered and an N-DATA ACKNOWLEDGE request issued by the user.

It is not a coincidence that the X.25 packet level protocol (X.25 PLP) can be used directly to provide a connection-oriented Network service. In fact, the specification of the Network service and the design of the current version (1984) of X.25 PLP have considerably influenced each other. The earlier (1980) version of the protocol is deficient, particularly regarding packet formats. As a consequence, it is unable to directly provide connection-oriented Network service, unless additional procedures are defined so that NS primitive parameters can be supported. In other words, a thin layer of protocol is required to be implemented by each end system that wishes to support connection-oriented Network service.

4.9 Inter-working Protocols

In this section we re-consider internetworking issues from the viewpoint of Network layer protocol. The design of a Network layer protocol is complex since the access protocols used over individual sub-networks may be different or may not fully support the Network service. As a consequence, an internet-work may require a sub-layer of converge protocol to be implemented to support end-to-end communication.

4.9.1 Introduction

A communication network may be formed by interconnecting two or more similar, or perhaps dissimilar, communication sub-networks. Although it is transparent to users, it is helpful to consider an internet-work as consisting of distinct sub-networks. It, thereby, enables one to study addressing, sub-network access protocols and their conversion, where necessary. While it is possible to directly interconnect the networks with identical sub-network access protocols to form one internet-work, we shall assume that the two sub-networks are connected using a network layer gateway, or an Inter-Working Unit (IWU) to be precise.

Figure 4.23 illustrates an internet-work of two sub-networks, connected using an IWU. The sub-network access protocols for the two sub-networks are not necessarily the same. It is important, at this stage, to verify whether the access protocols are rich enough to support all elements of the Network service. Provided, each access protocol is able to directly support the required Network service, the design of the network layer protocol may be simplified. Otherwise, additional procedures need to be defined in the form of a convergence protocol, as was the case with using the 1980 version of X.25 protocol.

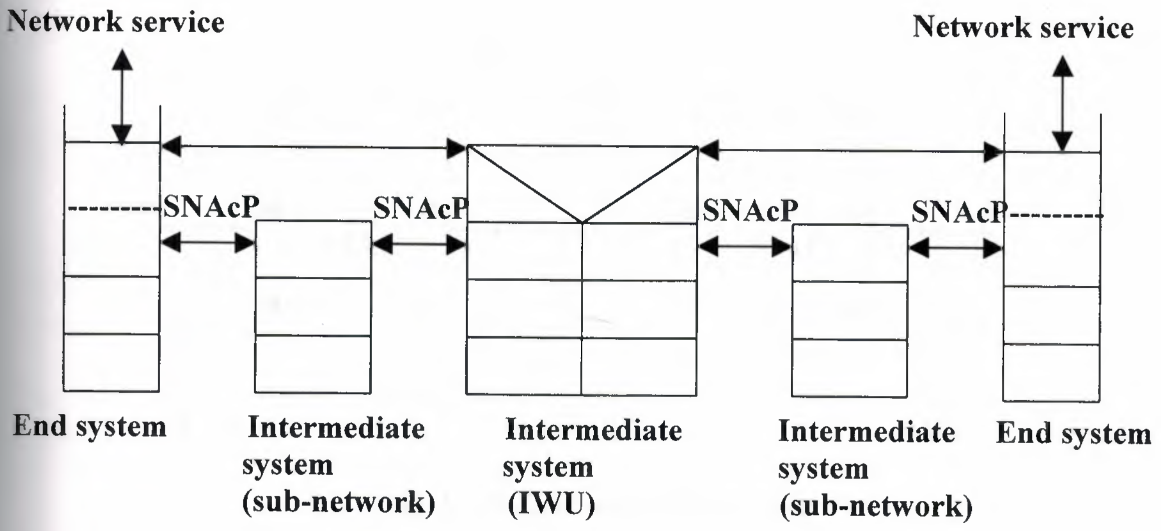
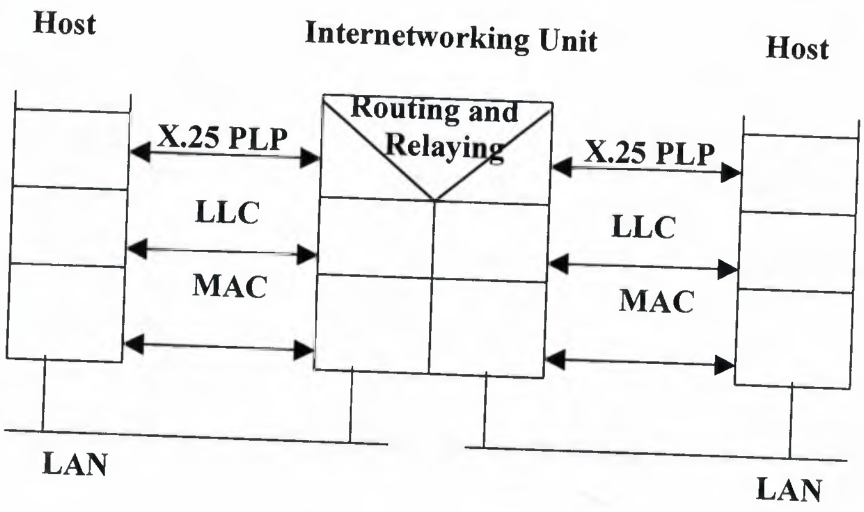

Network service
Network service
SNAcP
SNAcP
SNAcP
SNAcP
End system
Intermediate system (sub-network)
Intermediate system (IWU)
Intermediate system (sub-network)
End system

Figure 4.23 Internetworking of sub-networks using an Internetworking Unit.

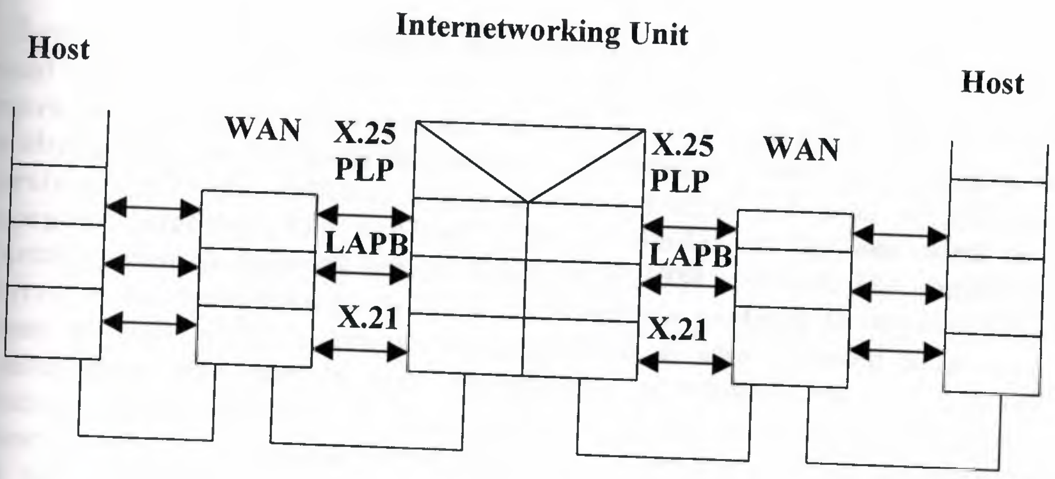
4.9.2 Interconnection of X.25 Networks

Below we discuss interconnection of sub-networks whose access protocols, though not identical, are able to support all elements of the Network service. In particular, we consider interconnection of X.25 (1984 version), as an access protocol, is capable of supporting connection-oriented Network service, and that it can be implemented over local or wide area networks. It is, therefore, to be expected that an interconnection of X.25 sub-networks using gateways could provide connection-oriented Network service to users in end systems. Figure 4.24 illustrates, respectively, LAN-LAN, WAN-WAN and LAN-WAN interconnections. Clearly, the protocol used at the physical and data link layers on the two sides of the IWU are independent of each other. Further, the operation of X.25 PLP protocols on the two side is also independent, except that as part of its relay function the IWU relays the events occurring on the sub-network to the other sub-network. For example, when a station on sub-network 1 needs to establish a Network connection with a machine connected to sub-network 2, an X.25 virtual call is established between a station and the IWU across each sub-network. The IWU (gateway), on its part, relays each event on a virtual call onto other hand. Thus from the viewpoint of NS users in end systems, the concatenation of two X.25 virtual calls appears as one X.25 virtual call which can then effectively support the end-to-end Network connection.

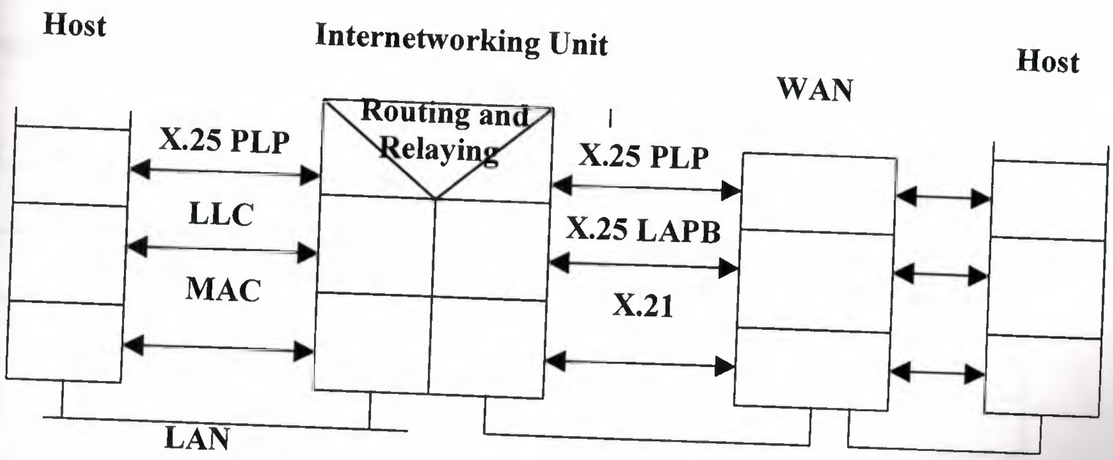
In the context of wide area public data networks, it is desirable to implement the IWU as two half-gateways, as illustrated in Figure 4.25. This, to some degree, solves the problem of distributed ownership and maintenance of gateways. In such cases, one may use CCITT's X.75 access protocol to link the two gateways. The X.75 protocol is very similar to X.25 access protocol, except that it is symmetric. Note that X.25 PLP protocol defines an interface between a DTE and a DCE, which is inherently asymmetric.



(a) LAN-LAN interconnection.



(b) WAN-WAN interconnection



(c) WAN-LAN interconnection.

Figure 4. 24 Interconnection of X.25 based LANs and WANs.

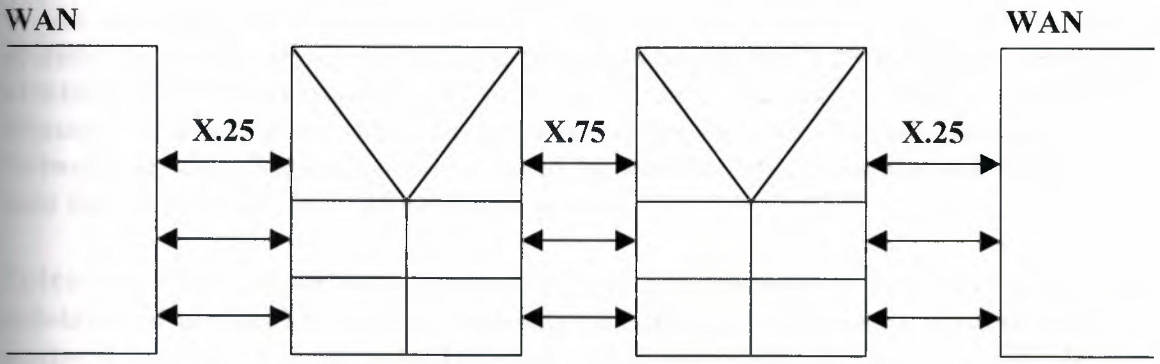


Figure 4.25 Interconnection of X.25 sub-networks using X.75 based half-gateways.

4.9.3 Converge Protocols

Hop-by-hop harmonisation. Consider now the case where the internet-work is formed using two sub-networks, but where the access protocol of at least one sub-network cannot support the required Network service. Figure 4.26 illustrates situation. One obvious approach to providing end-to-end Network service is to install a sub-layer of protocol over each deficient access protocol. Such a sub-layer of protocol, called convergence protocol. The approach here is similar to the one used to provide connection-oriented Network service using X.25 (1980 version). The context this time, however, is internetworking. As such the converge protocol is implemented by end systems as well as the gateway. The approach is also referred to as hop-by-hop harmonisation, and may be used irrespective of whether the sub-network access protocol is connection-oriented or connection-less, and whether the Network service is connection-oriented or connection-less. Note that the convergence protocols used over each sub-network are independent. It is the gateway which relays the semantics of events *from sub-network the another.*

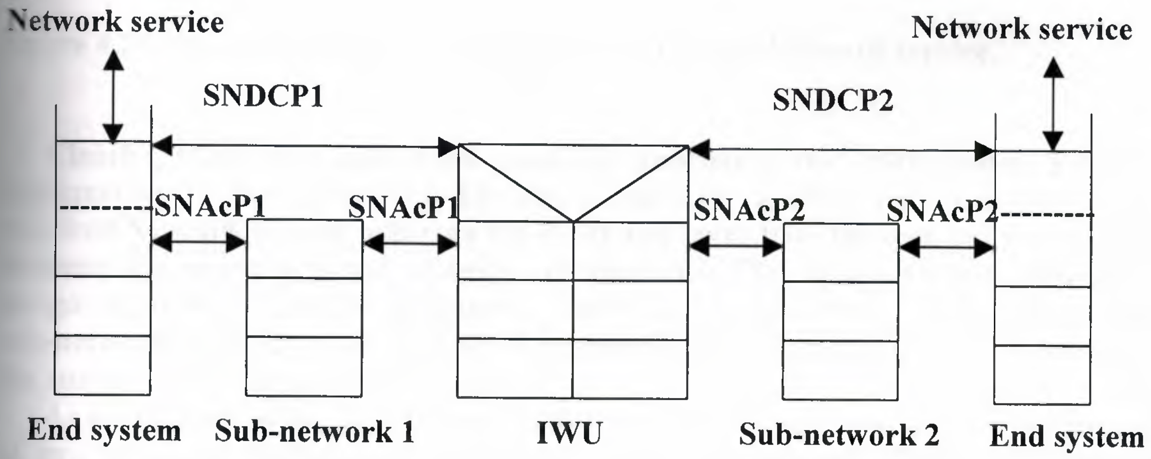


Figure 4.26 Hop-by-hop harmonisation: use of a convergence protocol across each sub-network to support Network service.

As one application of hop-by-hop harmonisation, consider providing connection-oriented Network service across an interconnection of twoX.25 based sub-networks, one of which uses 1980 version of X.25 protocol. Clearly, one may use the converge protocol, discussed earlier, over this sub-network to provide end-to-end connection-oriented Network service. No such protocol need be implemented over the sub-network which uses the 1984 version of X.25 recommendation.

Internet-work protocols. An alternative approach to internetworking requires that a sub-layer of protocol be defined and implemented across the entire internet-work. The protocol is called Sub-network Independent Convergence Protocol (or SNCIP), and is illustrated in Figure 4.27. Obviously, as the name implies, before such a protocol is defined and implemented, it must be ensured that the service available to it is independent of the access protocols of individual sub-networks, and uniform across the inter-work. In case an individual sub-network is unable to provide the required service element, then a Sub-network Dependent Convergence Protocol (or SNDCP) is implemented over and above its access protocol.

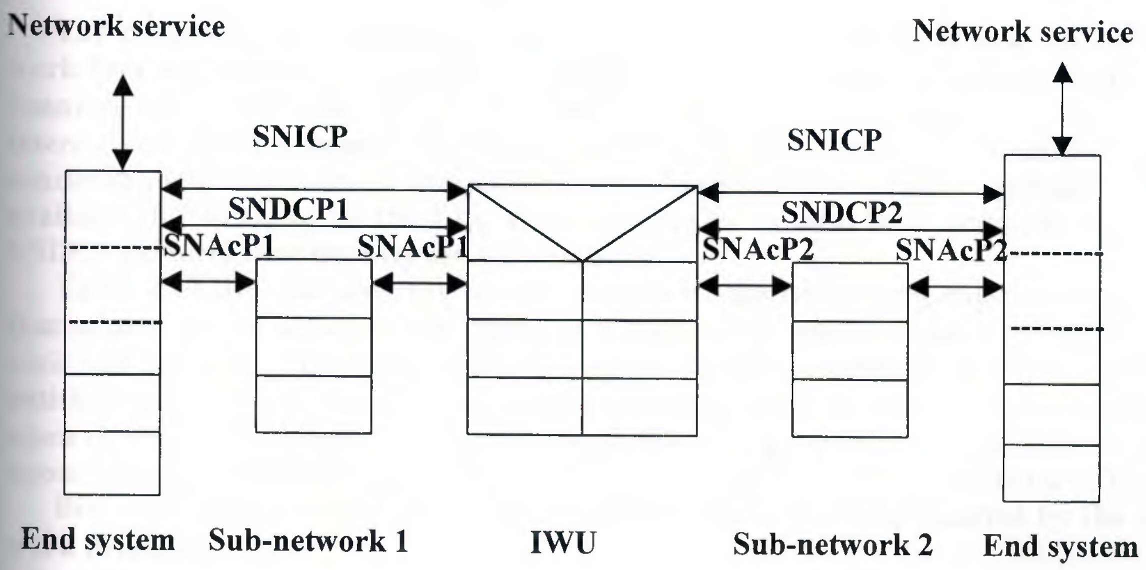


Figure 4.27 Use of an internet-work protocol to provide Network service.

Clearly, using the inter-work protocol approach, the convergence protocol is implemented in two sub-layers. The upper sub-layer is concerned with providing the required Network service, whereas the lower sub-layer tries the iron out the differences between the access protocol of each sub-networks. This approach also simplifies the design of gateways. That is, the gateway simply relays (and routes) the semantics across sub-networks with the sole purpose of supporting the intermediate service required by *the sub-network independent SNICP.*

As an example of the application of inter-work protocol, consider the interconnection of two networks, using an X.25 (1984 version) sub-network. figure 4.28 suggests a protocol stack that may be used to support connection-less Network service across the internet-work. The sub-network independent SNICP used across the internet-work is the Internet-work Protocol [ISO 8473], discussed later in this chapter.

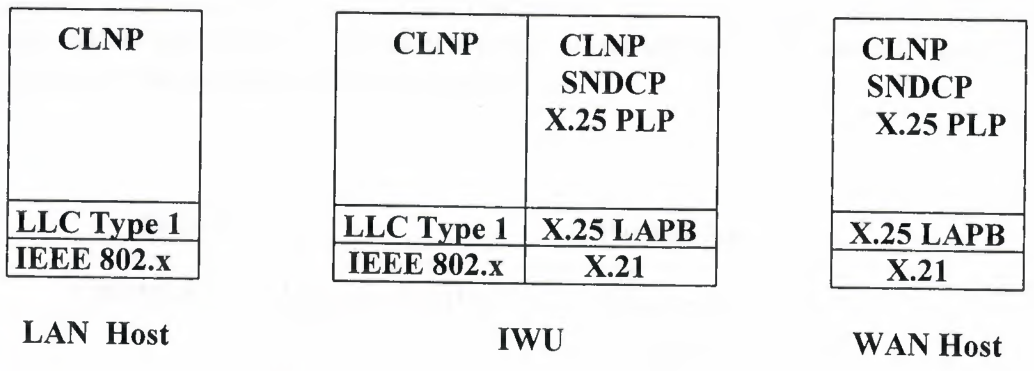


Figure 4.28 A suggested protocol stack for use over LAN-WAN interconnection.

4.9.4 Connection-less Network Protocol

The connection-less network protocol (or CLNP) is also referred to as an Internet-work Protocol, and is intended to be implemented by end systems to provide end-to-end connection-less Network service. It may be used across one sub-network, or an interconnection of a number of sub-networks. Further, it assumes the availability of connection-less data transfer service across each sub-network. such a capability is made available, for instance by the LLC (Class 1) service in local area networks, or by an SNDCP protocol running over an X.25 network.

Table 4.10 lists the only services it assumes of the underlying sub-networks. Note that addresses are sub-network points of attachments, whose significance is local the each sub-network. Therefore, it is the responsibility of intermediate IWUs (network entities, to be sure) to suitably route information through the inter-work, either based upon routing information that they generate from the given NSAP addresses, or based upon routing information already contained within the protocol data units (or PDUs).

Below we discuss some of the more significant functions implemented by the inter-work protocol:

1. *Lifetime Control of* PDUs: This function requires that a CLNP PDU be discarded by an intermediate Network entity, if it is known that it has been in the internet-work for a sufficiently long time. This feature helps to simply the design of a Transport layer protocol that ensures error-free connection-oriented data transfer across internet-work. The maximum lifetime of CLNP PDUs is determined using an estimate of the maximum end-to-end transfer delay.
2. *Segmentation of* PDUs: While the maximum size of user data in an N-UNITDATA request primitive is 64512 octets, it is rarely the case that underlying sub-network access protocol support such large PDUs. Therefore, the Internet-work protocol permits an NSDU to be transferred as a sequence of segmented PDUs with the same sequence number (or Data Unit Identifier). Since the segmented PDUs may be transferred through a number of intermediate sub-networks, each supporting a different maximum permissible PDU size, intermediate IWUs may further segment the received PDU segments. However, re-assembly of PDUs takes place only at the destination end system.

3. ***Routing of* PDUs: Once a CLNP PDU has been composed, the sending Network entity determines the next Network entity to which the PDU must be sent, as well as the underlying sub-network to be used.**

Primitives	Parameters
UNITDATA request/indication	Source SNPA Destination SNPA Quality of Service User Data

Table 4.10 Sub-network Service Primitives and Parameters

CHAPTER 5

The Transport Layer

While the Network layer, and those below, provide a path for data transfer between communicating end systems, the Transport layer is primarily responsible for providing end-to-end services and ensuring that such communication is largely error-free. In this chapter, we discuss the nature of Transport level services, and the variety of protocols necessary to bridge the gap between the services provided by the Network layer and those desired of the Transport layer. Connection-less Transport services and the required protocol are also discussed.

5.1 The Transport Layer

The section is an overview of the nature of services provided by the Transport layer. The different classes of protocols are also listed.

The Transport layer is situated between the Network layer and the Session layer (see Figure 5.1). While the Network layer spans the entire collection of open systems, the Transport layer, and those above, have components that are implemented only in end open systems, that is, systems where applications are implemented. This is so since all interactions within the Transport layer are end-to-end. Exchange of information between peer Transport entities is made possible by the end-to-end data transfer service provided by the Network layer. The services provided by the Transport later to user-entities is the Session layer are called Transport service. These services insure efficient and reliable data transfer between Session entities, independent of the underlying communication network or media (see Figure 5.1).

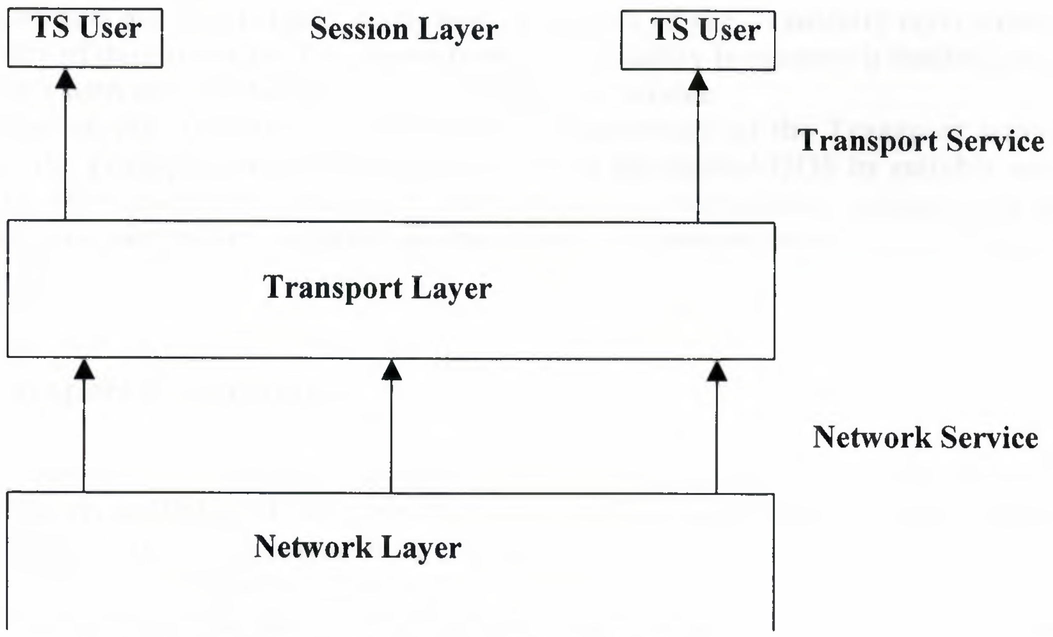


Figure 5.1 The Transport service, its users and the provider.

5.1.1 Data Transfer Characteristics

Elements of the Transport service may be classified into those that are connection-less services and those that are connection-oriented. In either case, data transfer service may be characterised as being:

1. end-to-end,
2. transparent,
3. independent of the underlying communication media,
4. varied in quality of service,
5. (possibly) reliable,
6. efficient or optimised.

Let us consider these characteristics.

1. End-to-end data transfer capability is largely derived from the Network service characteristics.
2. Transparency of information transfer refers to the fact the Transport layer places no constrains on the message contents or its coding.
3. Independence from the underlying communication media implies that the users of the Transport service do not experience a difference in the quality of service (or QOS) of Transport service as a result of changes in the Network service or its quality. The QOS of the Network service are substantially dependent on the transmission media used, as well as upon the networking technique employed.
4. The Transport layer may provide a variety of quality of service. Provision is thus made for Transport service users (or TS users) to request and negotiate, among themselves and with the Transport layer, the desired QOS of Transport service. The QOS may be characterised in terms of throughput, transit delay, residual error-rate, and failure probabilities.
5. Provision of reliable data transfer facility implies that data will be transferred error-free, loss-free, duplication-free, and possibly in the proper sequence. This functionally may sometimes be derived from the Network service. If not, functions and protocols are defined and implemented as part of the Transport layer achieve reliability of data transfer. The extent to which reliability is ensured is limited, but is consistent with the negotiated quality of Transport service.
6. Efficiency of data transfer is another major requirement of the Transport service. That is, the Transport service required to provide the desired QOS by suitably using available Network layer services and other resources. For instance, multiplexing and splitting are particularly relevant in the context of connection-oriented Transport service.

5.1.2 Transport Connections

In the context of Transport services, Session entities are its users, while the Transport layer, and those below, are provider of Transport service. These services made available by the TS provider at Transport-service-access-points (or TSAPs) to the attached TS users. Transport services may be classified into those that are connection-oriented or connection-less data transfer services, either or both of which may be offered by the Transport layer. Connection-oriented services assume that data transfer can begin only after a connection has been established between the TSAPs, to which the corresponding TS users are attached. Such a connection is referred to as Transport Connection, or TC.

A TC is established between two TSAPs on behalf of the attached TS users. Surely, there may exist a number of TCs at a TSAP. Further, there may even exist more that one TC between the same pair of TSAPs. The use of T-Connection-end-points (or TCEPs) allows one to distinguish between the various TCs established at the same TSAP. At a TCEP, the TS user and the supporting Transport entity refer to the corresponding TC using distinct TCEP-identifier. Thus, corresponding to each TC, there is an associated pair of TCEP-identifiers, one for each end of the TC. Note, these identifiers need not be the same, since their significance is only local.

5.1.3 Connection-Oriented Services

Connection-oriented Transport service includes service-elements to establish or release connections, or to transfer data transparently. Data is normally transferred in the form of Transport-data-service-units (or TDUs). These services are summarised below:

1. TC Establishment: A TS user may establish a TC with another TS user. The address of the TSAP to which corresponding TS user is attached is assumed to be known to the initiating TS user. During TC establishment, a TS user may request, and negotiate with the other TS user and the TS provider, the quality of service to be provided over the TC.
2. TC Release: Either TS user may unilaterally and unconditionally release the TC during its establishment, or subsequently. As one consequence, any data currently in transit may not be delivered, and destroyed.
3. Normal Data Transfer: TS users may exchange data, in the form of TSDUs consisting of an integral number of octets, such that the boundaries between TSDUs and their contents are preserved. The Transport layer may controls the rate at which a TS user sends octets of data sent, rather than the rate of TSDUs.
4. Expedited Data Transfer: A limited amount of user data may be transferred by a TS user in the form of Expedited-TSDU. But, this service being provider-optional, is available only if the TS users agree to use it and the TS provider agrees to provide it. Further, the transfer of Expedited-TSDUs may be subject to a similar, but distinct, flow control by the Transport layer.

5.1.4 Connection-less Services

Connection-less data transfer services, on the other hand, do not require the establishment of a connection prior to the data transfer. Thus, the only service-element available relates to Connection-less Data Transfer. A user may transparently transfer a TSDU, of restricted length, to a TS user. The address of the TSAP, to which the receiving TS user is attached, is known to the sending TS user. Each TSDU is sent independent of other TSDUs. While initiating the transfer, the sending TS user may request the desired quality of service that the TS provider must associate with the transfer.

5.1.5 Network Services Assumed

In order to provide Transport service, the Transport layer uses the available Network service. The available Network service quality may in some cases be comparable, or even identical, to the Transport service it provides. In that case, the design and the implementation of the Transport layer and its protocol is relatively simple. But when the Network layer provides minimum functionally, or is poor in quality, the Transport layer is fairly complex. In other words, the Transport layer implements those Transport functions that are necessary to bridge the gap between the Network service available to it and the Transport service which it offers.

5.2 Transport Protocols

Since there is a great variability in the Transport service to be provided and the Network service that may be available, a number of classes of Transport protocols are defined.

The responsibility of providing Transport services, described earlier, lies with the TS provider. Since its TS users are spread across open systems, implementation of a TS provider is in the form of a collection of co-operating Transport entities. Communication between Transport entities residing in different open systems must conform to a set of rules and procedures, so that there is no ambiguity in interpreting received messages. These rules are specified as a part of a Transport protocol. Further, communication between Transport entities requires that there be available some Network service, using which data units of the Transport protocol may exchanged.

From the viewpoint of standardisation, aspects of communication that are open and subject to standardisation are those that are related to services and to protocol only. A protocol standard specifies in detailed manner the semantics and syntax of all messages communicated between peer entities. Syntactical issues are important in the context of protocol specification, since a receiver must decipher from the received bits and bytes the message being encoded. The mapping of service primitives onto messages communicated is equally important, but only from the viewpoint of ensuring that a protocol achieves the goals of providing the defined Transport service(s). Similarly, specification of the manner in which a Transport entity uses the available Network service enables a common view of how Transport protocol data units are transferred.

5.2.1 Network Services

1. TC Establishment,
2. TC Release,
3. Normal Data Transfer, and optionally,
4. Expedited Data Transfer.

To provide these services, the Transport layer implements a number of functions that are necessary to bridge the gap between the Transport service it provides and the available Network service. Flow control, multiplexing, segmentation, error detection and recovery, and expedited data transfer are some example functions that may be implemented *by the Transport layer. The range of functions to be implemented depends* not only upon whether or not expedited data transfer service is to be provided, but also upon the availability of the certain optional Network layer services and their quality.

Connection-oriented NS

1. NC Establishment,
2. NC Release,
3. Normal Data Transfer,
4. NC Reset, and optionally,
5. Expedited Data Transfer,
6. Data Acknowledgement.

Connection-less NS

1. Connection-less Data Transfer.

It assumed that Expedited Data Transfer and Data Acknowledgement services are optionally provided, and that their use is negotiated in providing connection-oriented TS. Further, a protocol for connection-less TS does not make use of NC Reset, Data Acknowledgement, and Expedited Data transfer services.

5.2.2 Types of Network Connection

Frequency of signalled failures is similar to resilience. The difference is that only signalled NC release or reset are considered while computing the frequency of signalled failures. Based on this characterisation, an NC may be classified as one of the following types:

Type A: A network connection with an acceptably low residual error rate and acceptably low rate of signalled failures.

Type B: A network connection with an acceptably low residual error rate, but which has an unacceptably high rate of failures.

Type C: A network connection which has an unacceptably high residual error rate.

From the viewpoint of Transport layer protocol design, a high residual error rate is considered to be considered to be far more serious, thereby requiring a fairly complex protocol to carry out error detection and recovery. Thus, it is immaterial whether a Type C network connection has a low frequency of signalled failures or not. Further, the design of Transport protocol for use over a connection-less NS is likely to be as complex as the one that uses a Type C network connection, since they both exhibit similar error characteristics.

5.2.3 Protocol Classes

In view of the above classification of Network services, and the fact that Expedited Data service is optional, a variety of Transport protocols have been defined. These are:

1. protocols for providing connection-oriented Transport service using connection-oriented NS. Depending upon the type of connections available, either one or more of the following five classes of protocols may be implemented:
   (a) Class 0: Simple Class (TP0),
   (b) Class 1: Basic Error Recovery Class (TP1),
   (c) Class 2: Multiplexing Class (TP2),
   (d) Class 3: Error Recovery and Multiplexing Class (TP3),
   (e) Class 4: Error Detection and Recovery Class (TP4).
2. protocol for providing connection-oriented Transport service using a connection-less Network service.
3. Protocol for providing connection-less Transport service using connection-less or connection-oriented Network service.

5.2.4 Connection-less Transfer Protocol

This protocol specifies the procedures necessary to provide connection-less data transfer service between two TS users. The procedures, aimed at moving a user TSDU from one TSAP to another, are extremely simple, since the supporting Transport entities do not have to ensure reliable delivery of user data. Further, since connection-less data transfer service is on a Per TSDU basis, these procures for acknowledgement, flow control, multiplexing, error recovery, etc. are not relevant.

The connection-less data transfer protocol may either use the available connection-less Network service or connection-oriented Network service. As such the protocol defines two variations of a procedure to transfer Transport layer protocol-data-units, one for each type of available Network service. Error detection may optionally be carried out to enable a receiving Transport entity to detect, and thus discard, TPDUs that contain transmission errors.

5.3 Connection-Oriented Protocol

In the preceding section reference was made to a number of functions, for example, flow control, multiplexing, etc. But there are many more needed to implement the different classes of connection-oriented protocols.

With each function, there is an associated procedure which specifies the details of all communications that take place between peer Transport entities, and how such communication is affected using an available service. As to when a function is invoked, is not of particular concern here. There may be correspondence between issuing of TS primitives and invoking of these procedures. For instance, when a TS user issues a T-EXPEDITED DATA request primitive, the supporting Transport entity invokes the expedited data transfer procedure. On the other hand, multiplexing is used within the Transport layer to provide efficient and cost-effective data transfer service.

5.3.1 Transport-Protocol-Data-Units

Each procedure describes communication between Transport entities in terms of TPDUs exchanged between them. It specifies the contents of each TPDU and the interpretation that a receiving Transport entity associates with each TPDU and its parameters.

The collection of TPDUs required to implement each class of connection-oriented Transport protocols is listed Table 5.1. Note that some of the TPDUs are used only when certain options are negotiated in protocol Classes 1 and 2.

The following functions are commonly used in all classes of protocols that support connection-oriented TS and use connection-oriented NS.

1. Assignment to Network Connection,
2. Transfer of TPDUs,
3. Connection Establishment,
4. Connection Refusal,
5. Connection Release,
6. Association of TPDUs with TC,
7. Treatment of Protocol Errors,
8. Segmentation and Re-assembly.

Table 5.1 Applicable TPDUs for Each Protocol Class

TPDUs	Protocol Class				
	TP0	TP1	TP2	TP3	TP4
CR: Connection Request	*	*	*	*	*
CC: Connection Confirm	*	*	*	*	*
DR: Disconnect Request	*	*	*	*	*
DC: Disconnect Confirm		*	*	*	*
DT: Data	*	*	*	*	*
ED: Expedited Data		*	NF	*	*
AK: Data Acknowledgement		NRC	NF	*	*
EA: Expedited Data Acknowledgement		*	NF	*	*
RJ: Reject		*		*	
ER: TPDU Error	*	*	*	*	*

Note:
*: TPDU is always used,
NF: not available when non-explicit flow control is selected,
NRC: not available when receipt confirmation is selected.

5.3.2 Assignment to Network Connection

Each Transport connection is supported using a Network connection. That is, TPDUs concerning a TC are sent over an NC assigned to it. This assignment of a TC is made at the time of TC establishment. The procedure for Assignment to Network Connection enables a pair of communicating Transport entities to use the same NC to support all communication pertaining to a TC. The Transport entity which initiates te TC establishment procedure is the one responsible for assigning the TC to an NC. The assignment can, however, only be made an NC which the initiating Transport entity owns. That is, the NC to which the TC is assigned must have been established upon a request from the initiating entity. The responding Transport entity becomes aware of the assignment when it receives a TPDU requesting the establishment of a TC (that is, a CR TPDU) over the assigned NC.

5.3.3 Transfer of TPDUs

Each TPDU communicated between Transport entities is transferred over an NC, to which the TC is assigned, using Normal Data transfer or Expedited Data transfer or using N-DATA or N-EXPEDITED DATA primitives. In Class 1 protocol, expedited TS user data may be sent using the Expedited Data transfer service provided by the Network layer. This is, however, subject to availability and negotiation by the communicating Transport entities.

5.3.4 Connection Establishment

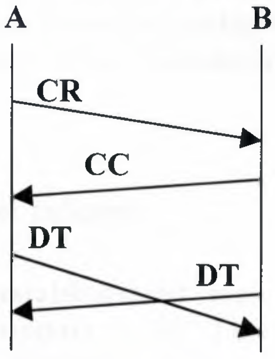
This procedures relates to the establishment of a Transport connection between a pair of supporting Transport entities. The initiating Transport entity sends a CR TPDU, to which the responding Transport entity responds with a CC TPDU if it accepts the establishment of the TC (see figure 5.2 (a), (b)). These TPDUs are sent using N-DATA primitives over the assigned NC. If, however, the responding Transport entity can not accept the connection, it responds with a DR TPDU, signifying a disconnection request.

A number of parameters are included in each of CR and CC TPDUs. Prominent among these are:

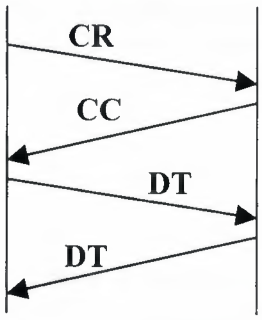
1. Calling and Called TSAP Addresses (optional, in case the NSAP Addresses uniquely identify TSAps),
2. Source and destination reference numbers, which are used to identify a Transport connection,
3. Initial Credit allocation in case flow control is used,
4. Proposed or selected values of negotiable parameters,
5. User data, if any.

Identify TC. A Transport connection is identified using a pair of reference numbers, one chosen by each communicating Transport entity. This identifier is, in fact, a Transport protocol-connection-identifier, and has significance which is local the communicating Transport entities. The initiating Transport entity chooses a source reference number (called SRC-REF), but assigns a value of 0 to DST-REF (a destination reference number) before sending the CR TPDU. The responding Transport entity chooses a value for the other reference number just before sending the CC TPDU. Subsequently, whenever a TPDU containing the parameters SRC-REF and/or DST-REF is sent, SRC-REF has the value of the reference number assigned by the sending Transport entity at the time of connection establishment. Similarly, DST-REF is assigned the value of the reference number chosen by the receiving (or destination) Transport entity.

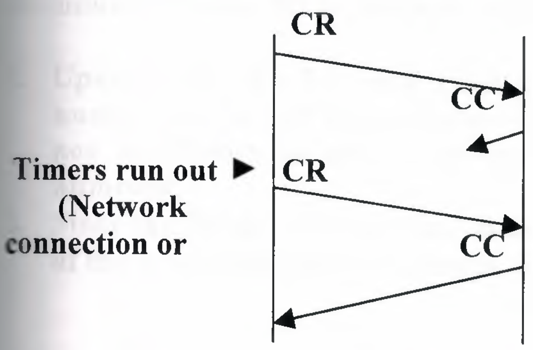
The range of values of reference numbers, and the mechanism for choosing one, is not specified by the protocols, except to limit its code to 16 bits. A reference number may be reassigned to another TC, one the TC has been released and it is reasonably clear that no TPDU concerning the released TC is anywhere in the network.



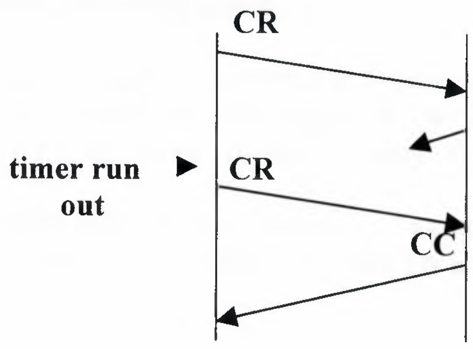
(a) Successful establishment in protocols classes 0 through 3



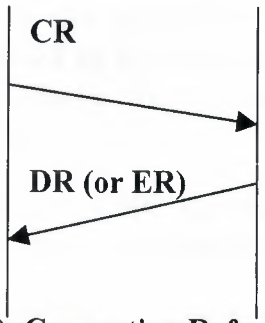
(b) Successful establishment in protocol class 4



(c) Multiple attempts at establishment in protocol classes 0 through 3



(d) Re-transmission of CR TPDU in protocol class 4



(e) Connection Refusal

Figure 5.2 Connection establishment.

Negotiation of protocol class and options. During connection establishment a number of parameters are negotiated, including:

1. protocol class,
2. use of optional functions of their variants, including window size, and use of expedited data transfer and checksums,
3. maximum TPDU size,
4. quality of service parameters, such as throughput and security.

The choice of protocol class is primarily dictated by TS user requirements in terms of optional services, quality of service, and by the type of available network connection. The Transport entity proposes a preferred protocol class and possibly some alternatives, while the responding Transport entity either totally rejects the connection establishment or selects a protocol class, depending upon its view of TS user requirements and available resources.

5.3.5 Connection release

A TC upon establishment may be released, in Class 0 protocol, by simply disconnecting the supporting NC. This procedure is referred to as the implicit variant of connection release. However, in protocol Classes 1 through 4 where either multiplexing is admissible or where error recovery is feasible, a TC release is initiated by a transport entity considers the connection as closed once it has responded with a DC TPDU, signifying Disconnection Confirmation. The supporting NC may, if necessary, be disconnected using NS primitives. Additional comments follow:

1. Upon release of a TC, each Transport entity may be required to freeze the reference number that it had earlier assigned to the TC. In protocol classes 0 and 2, where it is not mandatory to do so, an implementation may, in fact, freeze the reference numbers.
2. Since the TC may be released, either at the request of a TS user or because of failure of the supporting NC, the reason, if available, is made known to the TS user(s).

5.3.6 Association of TPDUs with TC

Once individual TPDUs have been separated out, their association with a TC is primarily based on a Transport protocol-connection-identifier. This identifier, as mentioned earlier in this section, is the paired reference numbers (SRC-REF, DST-REF). In fact, the DST-REF contained in the received TPDU is used, in most cases, to associate the TPDU with the TC. There are two additional difficulties in using this scheme. These relate to the following:

1. The TPDU may be received over an NC to which the corresponding TC is not currently assigned. In such a case, reassignment of TC onto a different NC is presumed.
2. At the time the TPDU is received, the TC may have already been released, but only from the viewpoint of the receiving Transport entity. In that case a TPDU is

returned that confirms the release of the corresponding TC, or the received TPDU is ignored.

5.4 Connection-less Protocol Procedures

In this section we discuss procedures that are relevant to this protocol.

5.4.1 Transport-Protocol-Data-Units

Only one type of TPDU is defined for connection-less Transport protocol, UD TPDU (for Unit-data). Such a TPDU primarily contains user data provided by a TS user in the form of the TSDU in a T-UNITDATA request primitive. Aside from the TSDU, the source and destination TSAP addresses, and optionally a Checksum parameter, are also included. The TSAP addresses are also provided as part of T-UNITDATA request primitive.

5.4.2 Transfer of TPDUs

A UD TPDU is transferred by a Transport entity to the relevant peer Transport entity using connection-less or connection-oriented data transfer service provided by the Network layer. This selection of Network service is based upon the availability of these services, and the quality of service requested by the TS user. Quality of service parameters include transit delay, protection from unauthorised access, cost, and residual error rate. A connection-less Network service is likely to offer better delay and cost characteristics while a connection-oriented Network service offers, in general, a smaller residual error rate.

In either case, the sending Transport entity uses it address mapping function to determine the source and destination NSAP addresses from the given TSAP addresses. Further, depending upon the requested quality of service, a Transport entity may or may not include a checksum parameter in the UD TPDU. If residual error rate is particular concern then the checksum parameter may be included. A receiving Transport entity discards the TPDU if it determines that the TPDU is erroneous. No positive (or negative) acknowledgement is sent by the receiving Transport entity, nor is there any confirmation provided by the supporting Network layer.

Using connection-less network service. In case connection-less data transfer service is used, the UD TPDU is sent as NS user data (or NSDU) using N-UNITDATA service primitives. If the size of the UD TPDU exceeds the acceptable maximum NSDU size, the sending Transport entity may abandon transfer of UD TPDU, altogether.

Using connection-oriented network service. A UD TPDU is sent as NS user data using N-DATA service primitives, but only after a Network connection has been established. If the attempt to establish a Network connection fails, or if it disconnects prior to issuing an N-DATA primitive request, then the transfer of UD TPDU is abandoned by the Transport entity. Once data transfer is complete, the supporting Network connection may be released by either Transport entity. The Transport entities communicating a UD TPDU, of course, have to need to reset the Network connection.

CHAPTER 6

The Session Layer

The main functionally of the Session layer is to provide Presentation entities with the means to organise exchange of data over a connection, to negotiate release of the connection, or to place synchronisation points in the stream of data. The latter enables users to structure their communication in the form of a serious of dialogue units, and the subsequently resynchronise data exchange in the event of errors. Synchronisation points also allow users to define an activity that may be interrupt and later resumed. These services, and the necessary protocols, are discussed in this chapter.

6.1 Introduction

The Session layer is suited between the Transport layer, and the Presentation layer. As with the Transport layer, subsystems corresponding to the Session layer are present only in those open systems where Application entities reside. As such, all interactions between Session entities are end-to-end, and are possible by the services provided by the Transport layer (see Figure 6.1).

The Session layer, together with the layers below, provides services, called Session service (SS), to its user entities in the Presentation layer, and thereby, to the Application entities. These services are accessible by a Presentation entity at a Session service-access-point (SSAP), to which it is attached. As such, a Presentation entity attached to an SSAP is also referred to as an SS user, while the Session layer, together with layers below, is called the SS provider. Further, the Session service connection-oriented. That is, two Presentation entities may exchange data only after a connection has been established between the SSAPs, to which they are, respectively, attached. Such a connection is called a Session connection.

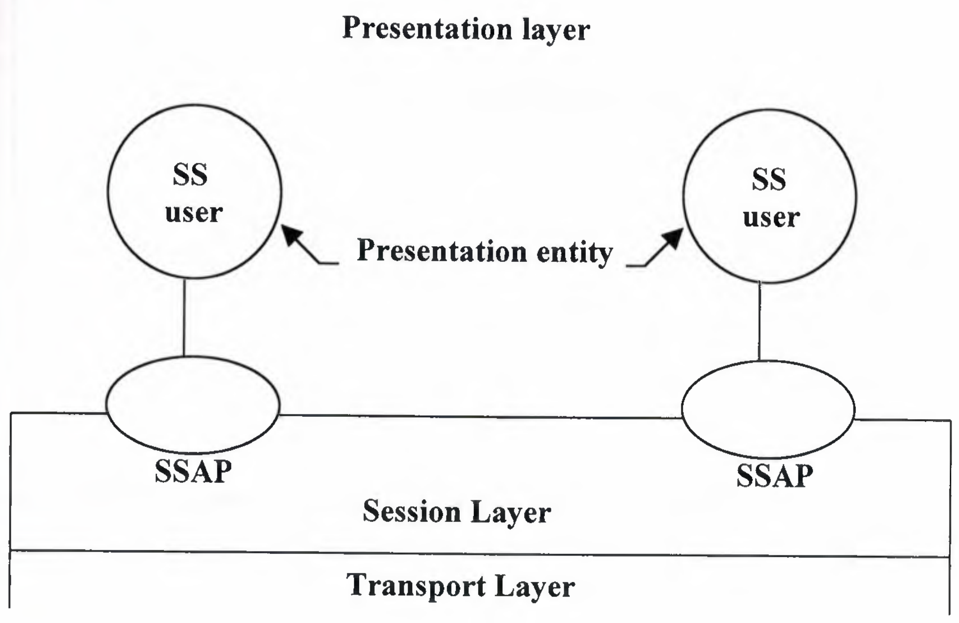


Figure 6.1 The Session layer.

6.1.1 Session Connections

A Session connection is established by the Session layer between two SSAPs on behalf on the attached Presentation entities. The Presentation entity requesting the establishment of a Session connection provides to the Session layer the address of the SSAP, to which the responding Presentation entity attached. Such an address may have been obtained using the directory or the address mapping function of the Presentation layer. Since there may exist a number of connections between an SSAP and other SSAPs, each Session connection is identified by its and point. Such an identifier is called Session-connection-end-point identifier (SCEP-identifier).Thus, for each connection there is an associated pair of SCEP-identifier, one for each connection there is an associated pair of SCEP-identifier, one for each end of the connection. An SCEP-identifier allows the Session service provider and the attached Presentation entity to uniquely identify (or refer to) the connection.(See Figure 6.2).

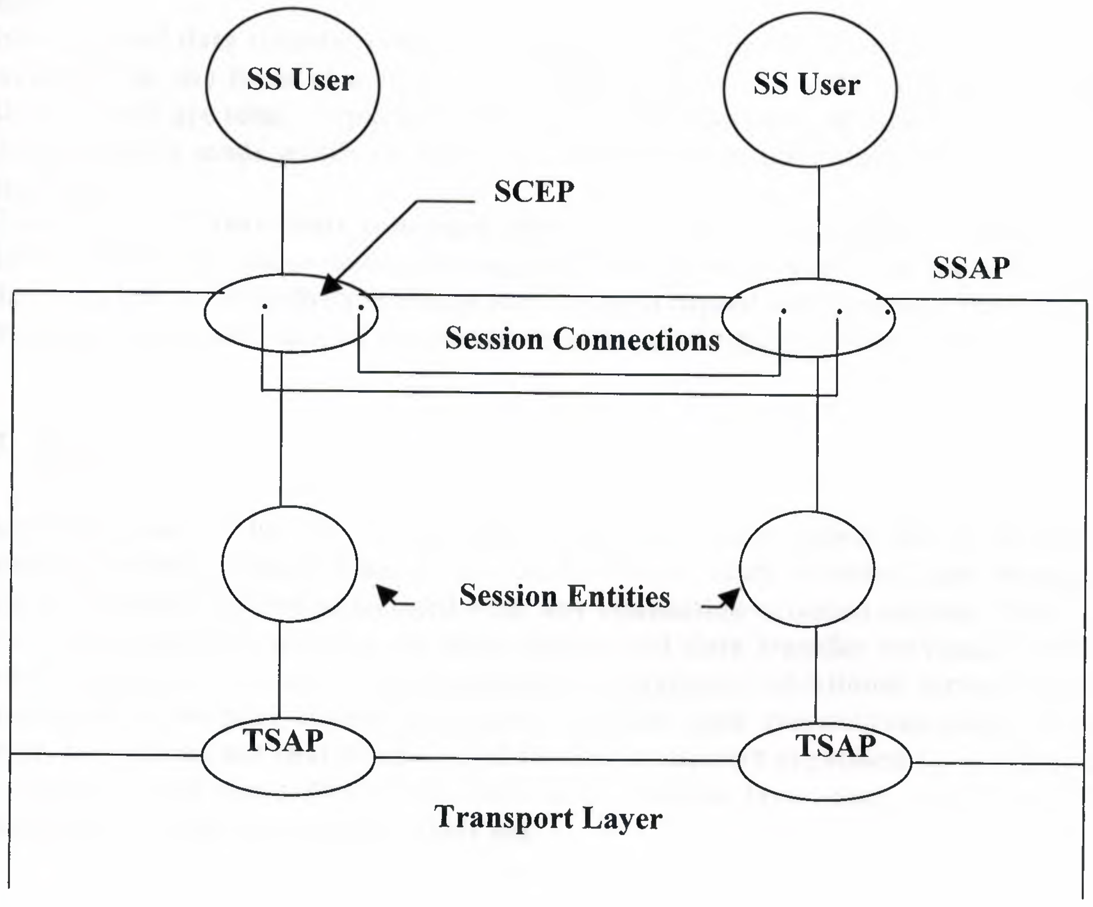


Figure 6.2 Session connections, SSAPs and SCEP identifiers.

6.1.2 Data Transfer Characteristics

Data transfer over a Session connection is end-to-end and reliable, a characteristic largely derived from the connection-oriented Transport service. Additionally, data is transferred transparently and independent of the underlying Transport connection is set up or maintained. However, the quality of data transfer service over a Session connection, in respect of throughput, delay, is to a great extent dependent upon the quality of Session service requested by an SS user determines the quality of service required of the Transport service. Further, a Session connection exhibits the following additional characteristics:

1. Interaction between two users over a Session connection may be organised. That is, the users co-operate between themselves to determine as to who may initiate certain operations over the connection at any given time. Operations that are subject to such control are half-duplex data transfer, orderly release, and synchronisation. As to how such a decision is arrived at is not the concern of the Session layer. The SS provider simply enables transfer of control over the connection from one to the other.
2. Synchronised data transfer refers to an ability on the part of users to structure their exchanges in the form of a series of a dialogue units. All data exchanges within a dialogue unit are totally separated from those that take place in other dialogue units. Using services made available unit, and resynchronise their data exchanges, when necessary.
3. Users may structure their communication in the form of an activity. Many of the data transfer operations have meaning only within an activity. The most important characteristic of an activity is that it may be interrupted and resumed subsequently, either during the life time of the current Session connection or a fresh one.

6.1.3 Services

Services provided by the Session layer may be broken down into a number of individual service elements. Connection establishment, data transfer, and connection release are similar to those associated with any connection-oriented service. There are, however, some differences in connection release and data transfer services, primarily since both these services are subject to being to organised. Additional service elements are required to support synchronised data transfer and resynchronisation. It may, however, be pointed out that provision of service to support organised or synchronised data transfer is not mandatory on the part of the Session layer, and, even if provided, the users may or may not negotiate their use.

6.1.4 Session Layer Protocol

The Session layer protocol implements those functions that allow SS user entities to organise and/or synchronise data transfer. As such, there is only one class of Session layer protocol, within which a number of options are available. These options pertain to various functional units, including half-duplex data transfer, synchronisation, negotiated release.

Functions of error detection and recovery, re-sequencing, flow control, are of little importance in the context of Session layer, since each Session connection is mapped onto a relatively error-free Transport connection. If Transport connection fails, or if a protocol is detected then the corresponding Session connection is aborted. It is for the SS users to re-establish a Session connection and resynchronise exchange of data. The latter would be feasible, provided an activity is in progress.

6.2 Organised and Synchronised Data Transfer

Organised data transfer refers to service provided by the Session layer, whereby a pair of SS users, in a co-operate manner, determine who may initiate certain operations related to a Session connection at a given time. These operations relate to data transfer, orderly release, synchronisation, and to activity management. These concepts, together with that of synchronisation, are discussed in this section.

6.2.1 Half Duplex Data Transfer

(Normal) data transfer may either be full duplex (two way simultaneous) or half duplex (two way alternative). In the latter case, at any time, at most one SS user has exclusive rights to initiate transfer of data over the connection. To enable SS users to transfer control over the connection, the notion of a token, or more precisely a data token, is defined.

1. If, at the time of connection establishment, half duplex data transfer has been negotiated, then the data token is said to be available.
2. Otherwise the token is not available, in which case there exist no constraint on an SS user issuing data transfer primitives. But, if the token is available, then, at any time, it is either.

In the former case, 2(a), only the user whom the data token is assigned, may initiate data transfer by issuing an S-DATA request primitive. The latter case, 2(b), may arise when the token is being transferred by a user to other. Figure 6.3 illustrates half duplex data transfer between SS users. Note that the service element S-DATA is unconfirmed.

Soon after a connection has been established, and if the data token is available, then it may be assigned by the initiating SS user to itself, or to the other SS user. The initiating SS user may, if it so chooses, leave it to the responding SS user to assign the data token. Subsequently, the SS user, to which the data token is currently assigned, may give the token to the corresponding user using the service element Give Tokens. Or, an SS user may even request the corresponding SS user to transfer the token to it using the service element Please Tokens. Exchange of data token between SS users, using the primitives S-TOKEN-GIVE and S-TOKEN-PLEASE, is shown in Figure 6.3.

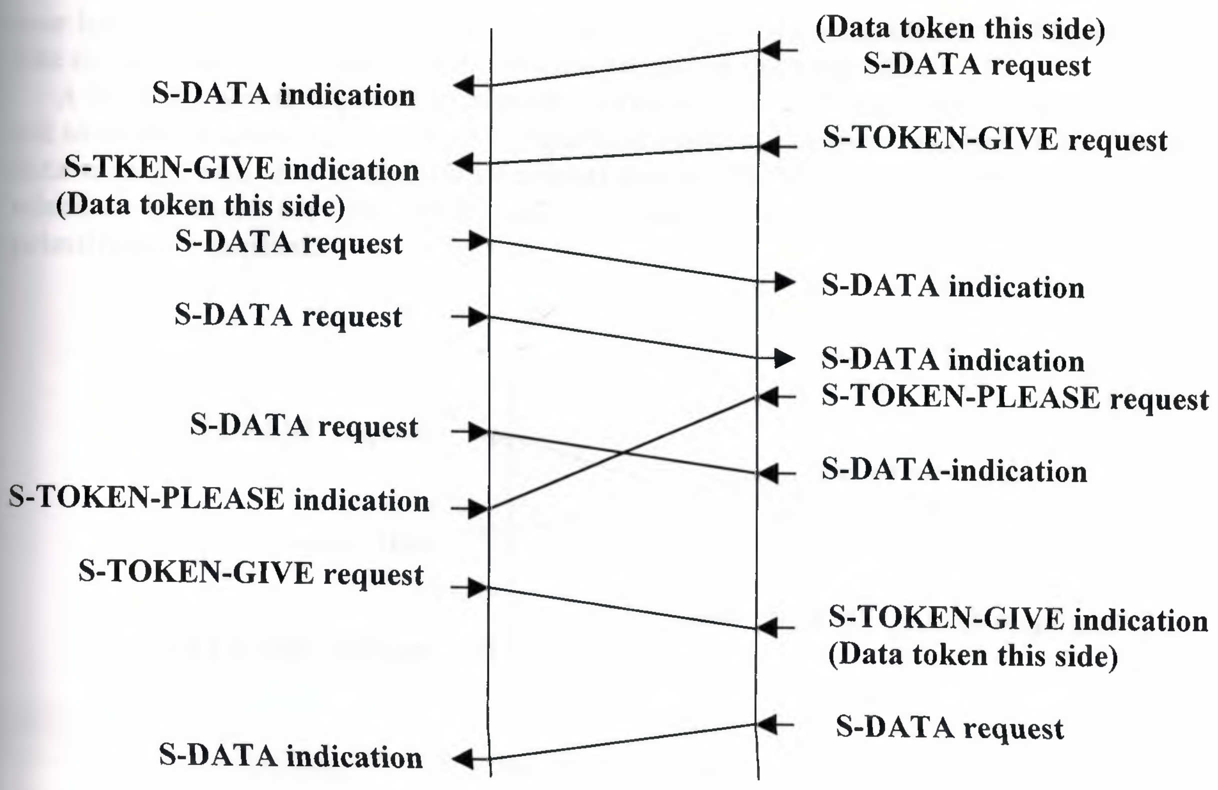


Figure 6.3 Half duplex data transfer and exchange of data tokens.

6.2.2 Negotiated Release

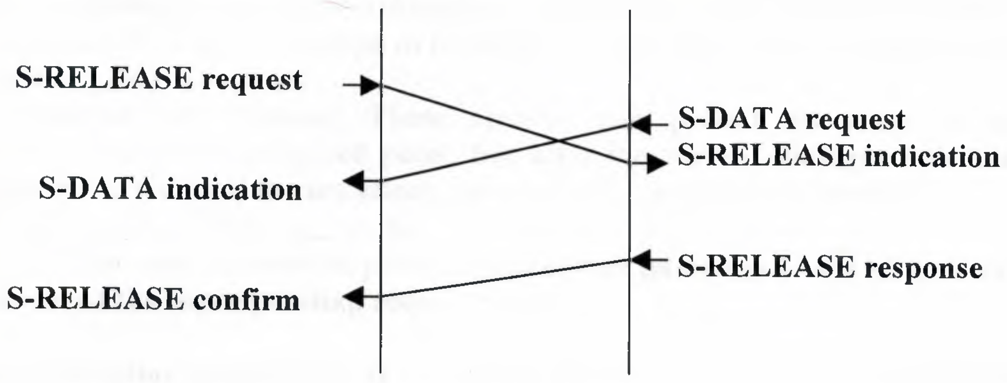
Yet another operation that is subject to being organised is connection release. These different connection release procedures are available. These are User Abort, Provider Abort, and Orderly Release. User Abort and Provider Abort are services elements, where an SS user or the SS provider may unilaterally abort the connection, respectively. The other parties involved are at best informed of its release. Note that, as a consequence, data in transit may be lost or destroyed.

Orderly release, on the other hand, involves a two way interaction between the two SS users, as well as the SS provider, before the connection is released. This ensures that data in transit is delivered before the connection is released. See Figure 6.4(a) for an illustration, where it is assumed that the service Orderly Released is confirmed, and that the associated service primitive is S-RELEASE. Orderly release has two important characteristics. These are:

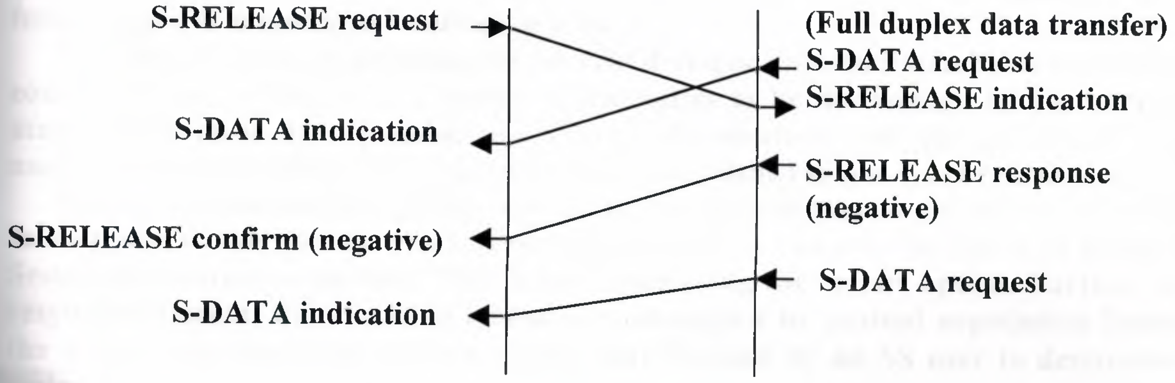
1. either SS user may initiate orderly release of the connection,
2. an SS user responding to the connection release has no option but to accept the release of the connection.

Instead, the SS users may agree at connection time to alter these characteristics of orderly release by making available a release token and, thereby, subject orderly release to be controlled by the current assignment of the release token. This form of connection release is called negotiated release, wherein the SS user, to which the release token is currently assigned, has exclusive right to release the connection. The corresponding SS user has, in that case, the right to refuse the release of the connection. See Figure 6.4(b). The management of release token is similar to that of the data token.

A Session layer is required to provide all three forms of disconnection. It may or may not implement functions to support negotiated release. Further, selection of half duplex data transfer and that of negotiated release are totally independent. But, for any given selection of these services, additional constraints on issuing corresponding service primitives are implied.



(a) Orderly Release.



(b) Negotiated Release

Figure 6.4 Release of a Session connection.

6.2.3 Resynchronisation

Although rare, the possibility of occurrence of an error, either within the Session layer or in logical communication between the two users, cannot be ruled out. Or, users may wish to restore the environment that may have existed earlier. To handle such requirements, SS users may initiate a resynchronisation procedure that allows them to set the state of their communication to a defined state. Such a state may be last confirmed major synchronisation point, or a subsequent minor synchronisation point. That is, issuing S-RESYNCHRONISE primitives, SS users may resynchronise to any synchronisation point within the current dialogue unit. In other cases, users may even define a new state, if the current dialogue units is to be abruptly terminated, and a new one started. As such, three different forms of resynchronisation are defined. These are referred to as options, and are discussed below:

1. The Restart Option: The restart option enables SS users to resynchronise communication to a synchronisation point, previously defined. Such a point may be the last major synchronisation point or a minor synchronisation point place within the current dialogue unit. While making the request, the initiating SS user identifies the serial number, s1, of the synchronisation point to which communication is to be resynchronised. s1 is constrained to be $V(R) \leq s1 \leq V(M)$. Once resynchronisation is complete, $V(M) \longleftarrow s1$.
2. The Abandon Set Options: These options also permit SS users to resume communication from an agreed point, but after the current dialogue unit has been terminated. In terms of the net effect, once resynchronisation is complete,
   $V(M) \longleftarrow s1, \quad V(R) \longleftarrow 0,$
   where s1 is the synchronisation point serial number passed by/to the SS user as a parameter of the corresponding request primitive.

If the abandon option is used, s1 is the next available serial number, provided by the SS provider, while the serial number is provided by an SS user in case of set or restart options. This state is similar to the one each once a connection has been established. In such respects, therefore, there does seem to be a similarly between the two options. The difference between the two is in respect of the serial number to be associated with a future major or minor synchronisation point.

Whether or not users abandon the current dialogue unit, and thereby disregarded all communication within it, is a matter of semantics to be determined by the users, by mutual agreement. One possible use of resynchronisation, with the set option, is for users to resynchronise to any of the previously established major synchronisation points.

With resynchronisation, data currently in the pipeline may be destroyed, or purged. In fact, one of the uses to which resynchronisation service may be put is to purge the Session connection of all data. This is best done using the restart option. Further, with resynchronisation, the available tokens are reassigned by mutual negotiation between the users. Also, resynchronisation service may be used by an SS user to destructively assign tokens to itself.

6.3 Session Protocol

Recall that the Transport layer simply provides services for end-to-end connection establishment, termination, normal data transfer and optionally expedited data transfer. Therefore, the Session layer is required to implement a variety of functions in order to close gap between Transport services and Session services. Aside from using a Transport connection to establish (or terminate) a Session connection, it must maintain state information concerning synchronisation points and activities and update the same each time a minor or major synchronisation point is placed, resynchronisation occurs, or an activity related service element is initiated. Further, the Session layer must ensure that interaction between SS users and the SS provider is consistent with the current distribution of available tokens.

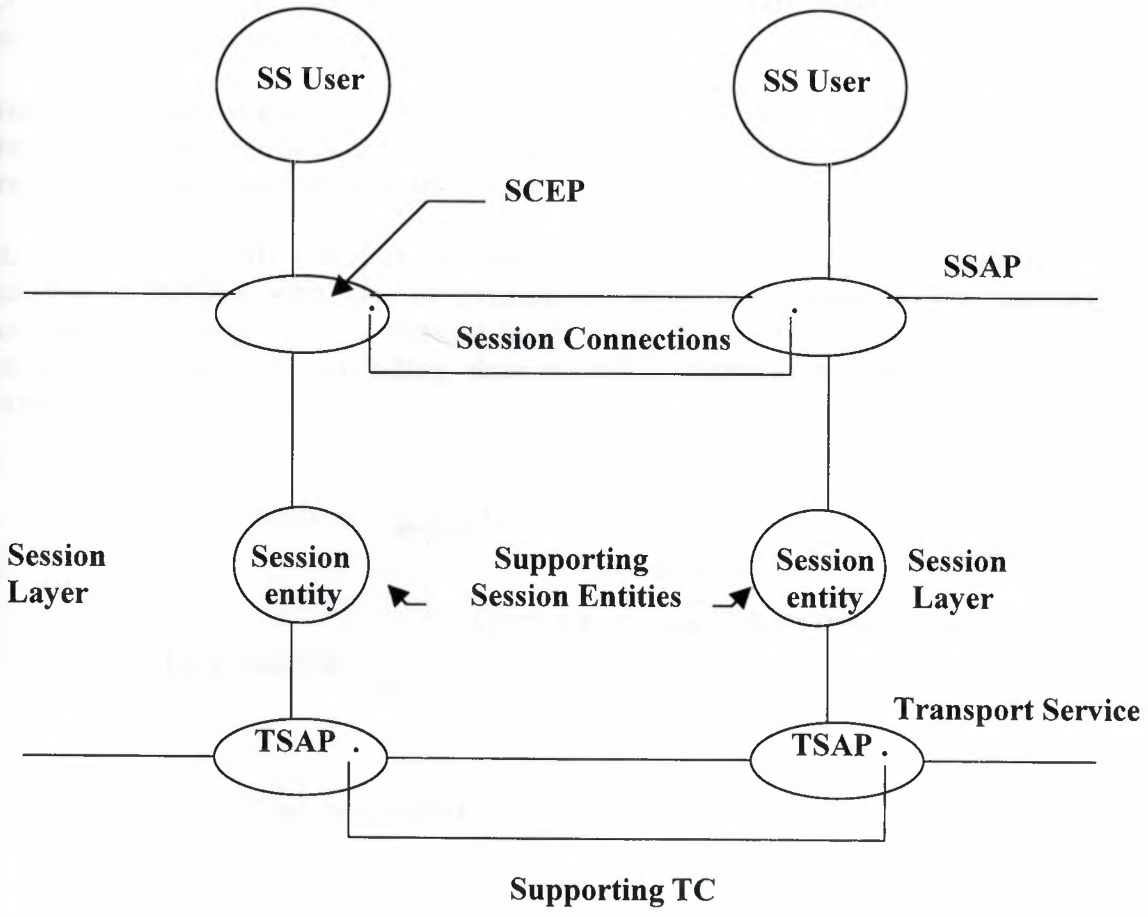


Figure 6.5 Model of the Session layer concerning a single Session connection.

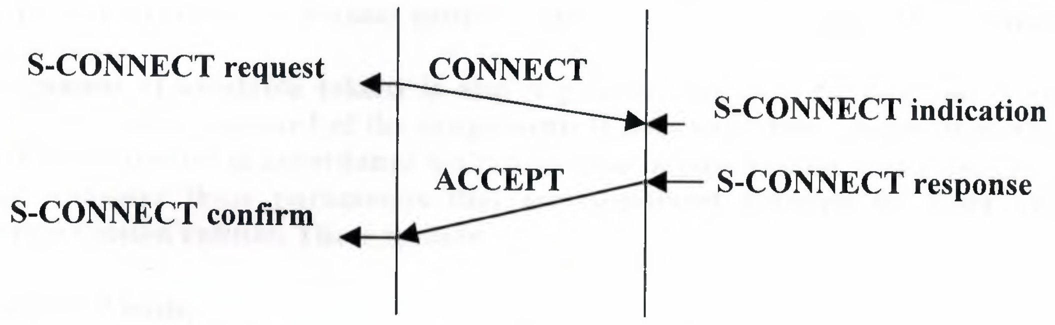
6.3.1 Session Protocol Data Units

Needless to say, that an important function of the Session layer is to convey the semantics of its interaction with an SS user over the connection to the other user. Figure 6.5 gives a model of the Session layer as it concerns a single Session connection. The interaction between a Session entity and the corresponding SS user is via service primitives. The two supporting Session entities interact with each other by exchanging Session Protocol Data Units (SPDUs) over a logical connection that they establish between themselves to support the particular connection between the two users.

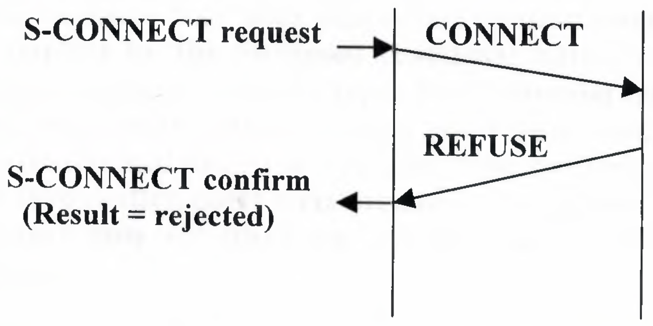
Exchange of SPDUs between the two entities is governed, both in respect of semantics and syntax, by the Session protocol. The protocol also specifies the event(s) that causes an SPDU to be sent, or the actions to be taken by a Session entity when it receives an SPDU. For instance, when an SS user issues an S-CONNECT request, its supporting Session entity, sends a CONNECT SPDU to the corresponding remote Session entity. Further, when the remote Session entity receives a CONNECT SPDU, it issues an S-CONNECT indication primitive to the corresponding SS user. The responding SS user's response is conveyed through an ACCEPT or REFUSE SPDU, depending upon whether the value of the Result parameter in the S-CONNECT response primitive is "accepted" or "rejected by SS user". Figure 6.6 illustrates the use of CONNECT, ACCEPT and REFUSE SPDUs in connection establishment. The figure illustrates three different cases corresponding to:

1. successful establishment,
2. rejection by the responding Session entity,
3. rejection by the responding Session user.

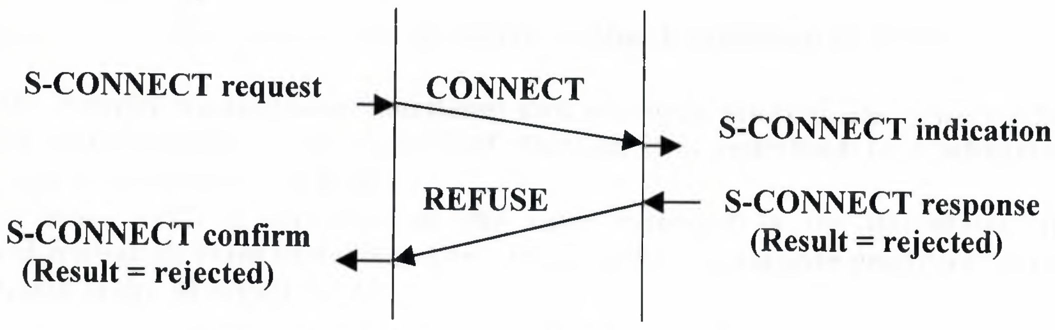
Table 6.2 lists the SPDUs that are associated with each Session service primitive. This association of SPDUs with service primitives immediately suggests the sequence of events that take place when a Session layer itself. Some of the other procedures are illustrates in Figure 6.7, including data transfer, resynchronisation, and provider-initiated abort.



(a) Successful connection establishment.



(b) Connection refused by responding Session entity.



(c) Connection refused by responding SS user.

Figure 6.6 Use of CONNECT, ACCEPT, REFUSE SPDUs connection establishment.

6.3.2 Connection Initialisation

During connection establishment, several aspects concerning procedures to be used subsequently are negotiated. Depending upon the procedure, the negotiation may take place between the two SS users alone, the two supporting Session entities, or between all four entities. We shall first consider:

1. assignment of Session Connection Identifier,
2. assignment of Activity Identifier, where applicable,
3. initial assignment of available tokens.

As noted earlier, Session Connection and Activity Identifiers are totally transparent to the SS provider (that is, to Session entities), and are, therefore, negotiated between the two users alone.

Assignment of available tokens is also negotiated between the two users, but the Session layer makes a record of the assignment. It does so to later ensure that SS users issue related primitive in accordance with constrains imposed by the token assignment.

Now, consider those parameters that are negotiated between SS users and the supporting Session entities. These include:

1. Functional units,
2. Initial Synchronisation Point Serial Number.

User requirements of functional units are purposed by the initiating SS user in parameter Session Requirements of S-CONNECT request primitive. The Session protocol assumes that both supporting Session entities are capable of providing the series implied by the purposed functional units. In other words, the initiating and responding entities are free to reject the connection establishment, in case either of them does not implement related functions (see Figure 6.6(b)). The applicable set of functional units is determined by the intersection of the two proposals.

Synchronisation point serial number, or equivalently the value of variable V(M), has significance only for when one or more of the following functional units has been negotiated:

1. Minor and/or Major synchronisation,
2. Resynchronisation,
3. Activity management.

In that case, the Synchronisation Point Serial Number is initialised as follows:

1. If the Activity management functional unit has been selected, the synchronisation point serial number is not initialised. Instead, it is initialised to 1 whenever an activity is "started" (see Table 6.1).
2. Otherwise, V(M) is initialised to the value returned in the parameter, Initial Synchronisation Point Serial Number, of S-CONNECT response primitive, or for the initiator in the ACCEPT SPDU.

Service elements	SPDU associated with request indication primitives	Associated ACK or reject SPDU
S-CONNECT	CONNECT	ACCEPT or REFUSE
S-RELEASE	FINISH	DISCONNECT or NOT FINISHED
S-U-ABORT	ABORT	-
S-P-ABORT	ABORT	-
S-TOKEN-GIVE	GIVE TOKENS	-
S-TOKEN-PLEASE	PLESE TOKENS	-
S-CONTROL-GIVE	GIVE TOKENS CONFIRM	-
S-DATA	DATA	-
S-EXPEDITED-DATA	EXPEDITED DATA	-
S-TYPED-DATA	TYPED DATA	-
S-CAPABILITY-DATA	CAPABILITY DATA	CAPABILITY DATA ACK
S-SYNC-MINOR	MINOR SYNC POINT	MINOR SYNC ACK
S-SYNC-MAJOR	MAJOR SYNC POINT	MAJOR SYNC ACK
S-RESYNCHRONISE	RESYNCHRONISE	RESYNCHRONISE ACK
S-U-EXCEPTION-REPORT	EXCEPTION DATA	-
S-P-EXCEPTION-REPORT	EXCEPTION REPORT	-
S-ACTIVITY-START	ACTIVITY START	-
S-ACTIVITY-RESUME	ACTIVITY RESUME	-
S-ACTIVITY-INTERRUPT	ACTIVITY INTERRUPT	ACTIVITY INTERRUPT ACK
S-ACTIVITY-DISCARD	ACTIVITY DISCARD	ACTIVITY DISCARD ACK
S-ACTIVITY-END	ACTIVITY END	ACTIVITY END ACK

Table 6.1 Mapping of Session service Primitives onto SPDUs.

Negotiation of the particular Session protocol to be used is clearly of concern only to the supporting Session entities. Therefore, each Session entity sends, to its peer entity, a list of protocol versions that it is capable of supporting. The negotiated version is the one with the largest version number present n the two lists. Currently versions 1 and 2 of the Session protocol are available. These versions refer to the 1984 and 1988 versions of CCITT's Session protocol.

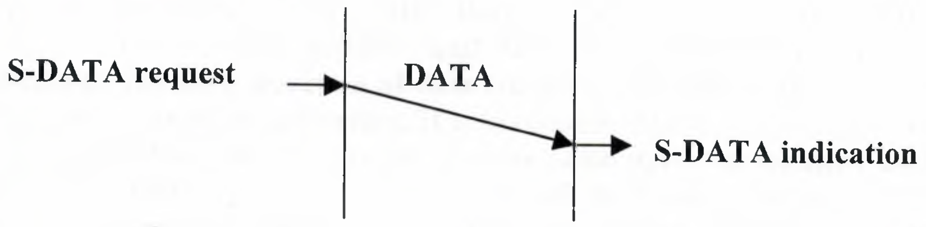
Session entities also determine themselves the manner in which an underlying Transport connection is used to support the Session connection. At any time, there is a one-to-one correspondence between the Session connection and supporting Transport connection. In other words, multiplexing or splitting functions are not used by the Session layer protocol. Instead, a Transport connection may be used to support another Session connection, but only after the former Session connection has been terminated. Thus, at the time of Session connection establishment, it is not necessary to establish afresh a Transport connection if there already exists a connection. Such a connection

1. is between a pair of TSAPs, to which the respective Session entities are attached,
2. the quality of service of the available Transport connection is comparable to those requested by the initiating SS user.

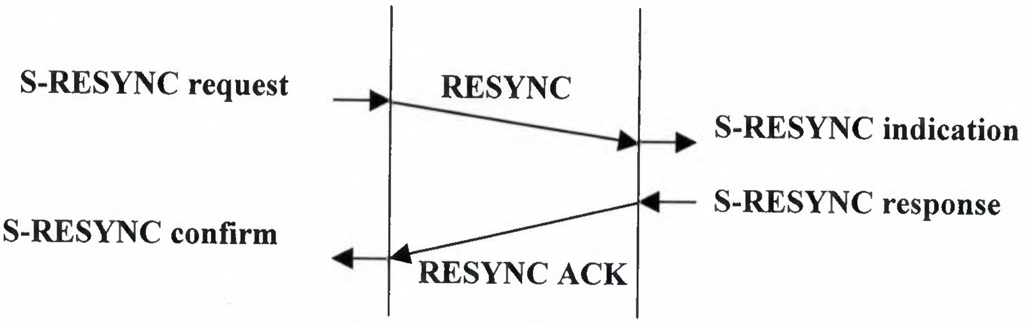
The Addresses of the Called TSAP is obtained using the address mapping function of the Session layer. This function also provides a Called Session Selector, used by the responding Session entity to uniquely identify the particular SSAP within the domain the Called TSAP. A similar procedure is used to obtain, from the given Calling SSAP Address, the Calling TSAP Address and the Calling Session Selector. While the Calling and Called TSAP Addresses are used to establish a supporting Transport connection, the Calling and Called Session Selectors are communicated as parameters in the CONCEPT SPDU.

There are other characteristics of the Transport connection that are of particular relevance to the Session protocol. These include

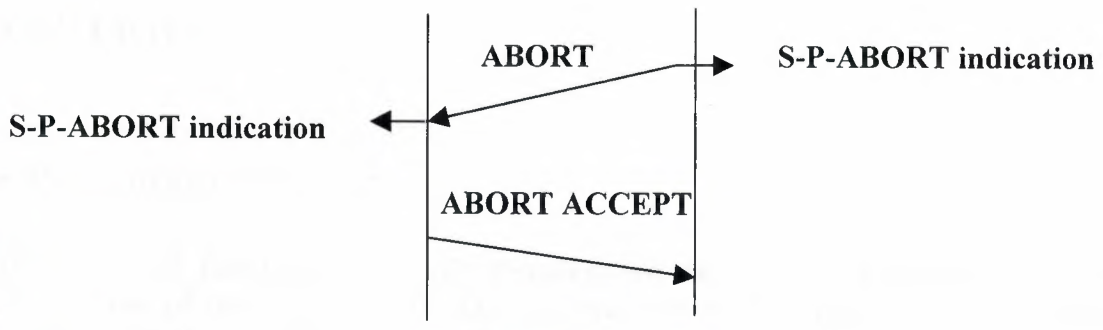
1. availability of Expedited Data transfer service over the Transport layer,
2. the maximum TSDU size.



(a) User initiated data transfer.



(b) Resynchronisation.



(c) Provider initiated abort.

Figure 6.7 Sequence of SPDU transmissions: some examples.

6.3.3 Use of Available Transport Service

Since the Transport service is limited to connection establishment, termination, and normal and expedited data transfer, and since the lifetime of the underlying Transport connection may be beyond that of the Session connection, it is evident that all SPDUs concerning the particular Session connection are sent as normal TSDUs. In case of Expedited Data transfer over the Transport connection is negotiated, then ABORT and ABORT ACCEPT SPDUs are sent as Expedited TSDUs.

As one consequence of the above, it may be necessary to negotiate an appropriate value of the maximum size of TSDUs that can be supported. Its negotiation is clearly, between the two Session entities and the TS provider. The negotiation is done independently for each direction of data transfer, and follows the procedure laid out by the Transport service specification. If a maximum TSDU size is negotiated successfully then, by implication, the two Session entities also agree to segment an SSDU (Session Service Data Unit), provided the length of its corresponding SPDU exceeds the negotiated maximum TSDU size. Obviously, each SPDU so constructed after segmentation must contain an indication of whether it contains the first, last or an intermediate segment of the corresponding SSDU. SSDUs from S-DATA, S-TYPED DATA, and other primitives are all subject to segmentation.

CHAPTER 7

The Presentation Layer

The main of functionally of the Presentation layer is to provide for suitable transformation of the syntax of all data exchange between Application layer entities. This ensures that the data exchanged can be interpreted appropriately by the two Application layer entity to represent information using a local syntax.

7.1 Introduction

This section is an introduction to services provided by the Presentation layer. In particular it covers issues that concern representation of information exchanged between Application entities. Presentation layer protocol is also briefly discussed.

7.1.1 Representation of Information

Recall from earlier chapters, that issues concerning end-to-end, reliable and cost-effective data transfer have already been resolved at the Transport layer. The Session layer ensures that such transfer is organised and/or synchronised. As such, major issues concerning data transfer have already been taken up and resolved by the Session layer, and those below. What remains to be discussed is representation of information exchanged between Application entities.

All along, data exchanged between users of given service has taken the for of a string of bits (or octets). But, given the fact that the users exchange a variety of information, the problem of representing user data is non-trivial. Either each Application entity itself encodes user information in a manner that is well understood by its peer entity, or this problem is solved by placing a separate layer – the Presentation layer – between the Session layer and the Application layer. The latter approach is adopted by the architecture of Open Systems Interconnection.

The Presentation layer, together with the lower layers, provides Presentation Service (PS). Using this service Application entities may transfer information without concern for how information is represented. This illustrated in Figure 7.1, where it is shown that two communicating Application entities, sometimes referred to as PS users, access these services at Presentation service access points (PSAPs) to which they are attached. The sending Application entity hands over user information to the Presentation layer using a representation scheme that is local to its interface with the Presentation layer. (The Presentation layer, together with the layers below, is referred to as PS provider.) The PS provider ensures that the user information is made available to the corresponding PS user using a representation that is also local to its interface, and possibly different from the former.

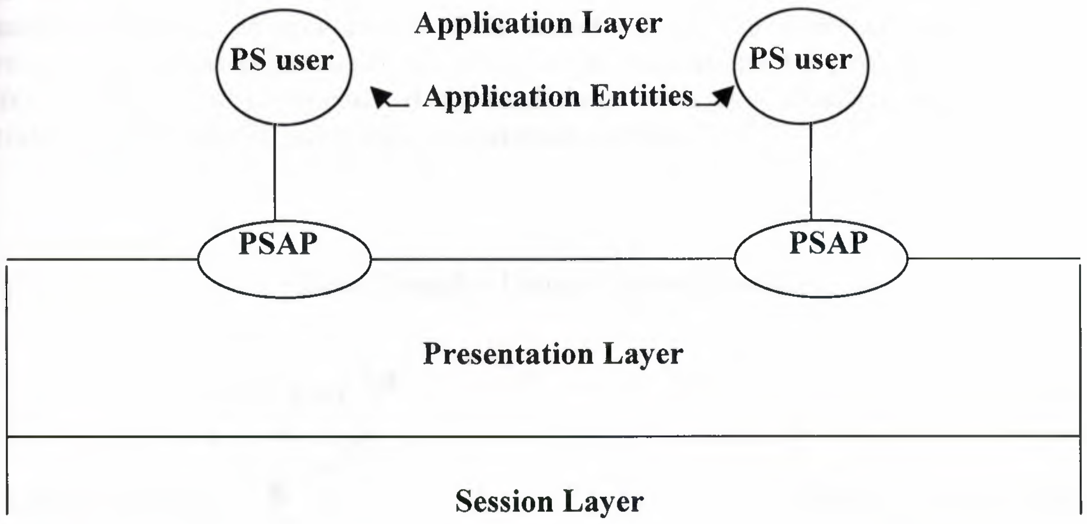


Figure 7.1 The Presentation Service Provider and its users.

7.1.2 The Abstract Syntax

Aside from transferring information from one user to the other, the Presentation layer carries out translation from one scheme of representation another. To do so, it must clearly be aware of the structure of user information. Obviously, this structure must also be known to the two communicating users for them to be able to associate a definite meaning with the information communicated.

Thus, the structure of user information must be unambiguously defined, and make known to the two users as well as the PS provider. The structure of user information may be either simple, for instance, an integer, a character, or an octet. Or, it may be more complex, in that it is defined in terms of a number of components each of which is either simple or complex, for example, a string of characters, a record of values, an or a sequence of records. The structure of user information is known as abstract syntax, and may be defined using the Abstract Syntax Notation One (ASN.1), which itself is a standard.

7.1.3 The Transfer Syntax

Above, we have made no mention of how the PS provider encodes user information and then transfers it from one user to the other. Figure 7.2 is a model of the Presentation layer, where it is shown that the supporting Presentation entity encodes the information into a string of bits (octets) using a particular scheme and transfers the coded string of bits to its peer Presentation entity. The scheme used to encode user information must be known to its peer Presentation entity as well. The Presentation entity, which receives the encoded user information, may decode the information content and hand it over to the corresponding user. The encoding scheme is known as the transfer syntax. Together, the abstract syntax and the transfer syntax are known as the presentation context.

Schemes for encoding user information whose structure is defined using the Abstract Syntax Notation have been standardised, and are available in reference.

To summarise the discussion thus far, user information that needs to be communicated must be structured. This structure, or the abstract syntax, must be known to, or negotiated between, the PS users as well as the PS provider. Further, the transfer syntax used to encode data consistent with the abstract syntax must be negotiated between the supporting Presentation entities.

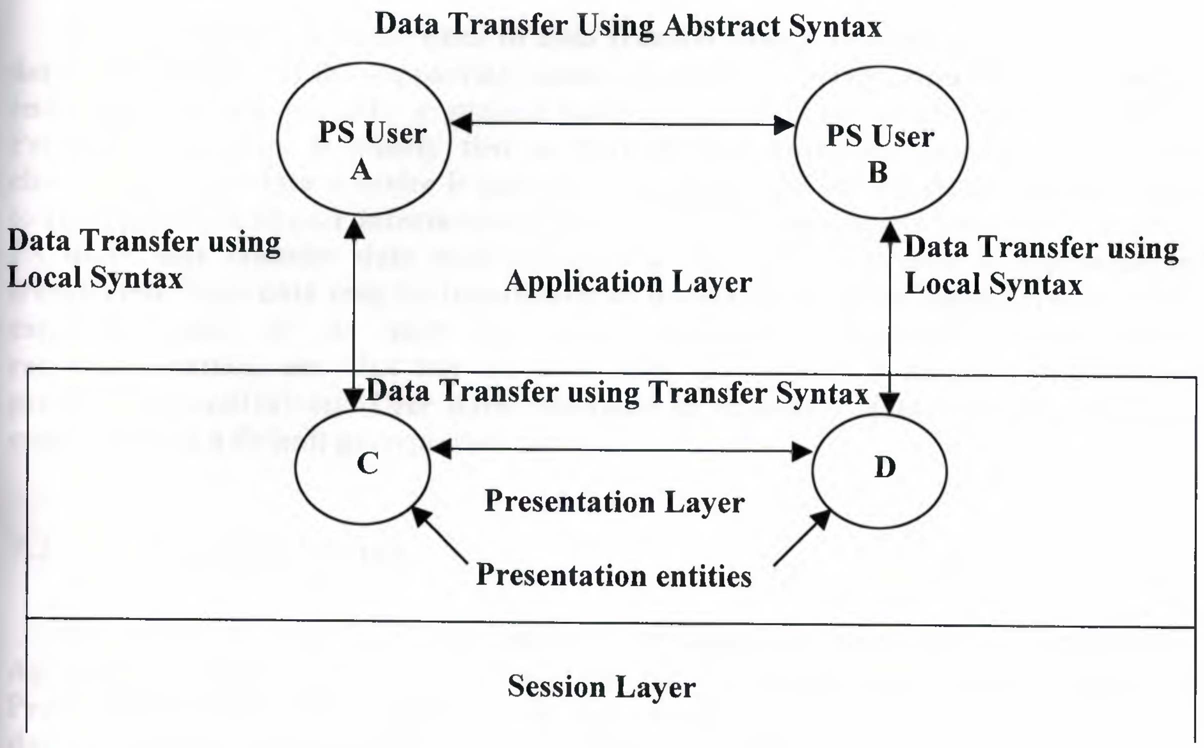

Data Transfer Using Abstract Syntax
PS User A
PS User B
Data Transfer using Local Syntax
Application Layer
Data Transfer using Local Syntax
Data Transfer using Transfer Syntax
C
D
Presentation Layer
Presentation entities
Session Layer

Figure 7.2 Data transfer using presentation services.

7.1.4 Presentation Services Characteristics

The Presentation layer provides services by which a presentation context can be negotiated between the concerned parties. This usually takes place at the time of connection establishment. In fact, at connection establishment, the concerned parties negotiate a set of presentation context. Furthermore, using available Presentation services it is possible for PS users to dynamically change the set of defined presentation contexts that are currently active. In case the presentation context set is empty, then a default context is used to encode user information. The default context also be at negotiated connection establishment. In the absence of any such negotiation, a default context is always available, which is known a-priori to all communicating Presentation entities and Presentation service users.

The above implies that the Presentation service is connection oriented. That is, Application entities may exchange information only after a connection has been established between PSAPs to which they are respectively attached. Such a connection is called a Presentation connection. Between a given pair of PSAPs, they may exist zero, one or more Presentation connections any time. Such connections may be distinguished from each other by associating with each connection a pair of Presentation-connection-end-point-identifiers (PCEP-identifiers), one for each end of the connection. The significance of a PCEP-identifier is, as usual, local. As such, the two PCEP-identifiers of a connection may or may not be distinct.

7.1.5 Data Transfer Characteristics

The majority of characteristics of data transfer over a Presentation connection are derived from those of the supporting Session connection. Surely, data transfer is end-to-end, reliable, and possibly organised and/or synchronised. Further, the quality of Presentation service is closely tied to that of the supporting Session service. The characteristics of data transfer is specially contributed by the Presentation layer relate to representation of user information. Once a presentation context has been negotiated, PS users may transfer data without concern for how such data is represent and transferred. User data may be transferred as normal data, typed data, expedited data, capability data or as part of certain operations including abort, release, resynchronisation, etc. But not all user data is subject to the currently defined presentation context set. User data contained in expedited data transfers, is always encoded using a default presentation context.

7.2 The Transfer Syntax

The abstract syntax of information exchanged between two communicating Application entities must be made known to the Presentation layer, so that supporting Presentation entities may appropriately determine an encoding scheme to represent it during transfer. The encoding algorithm, also known the Transfer Syntax, is negotiated only between the Presentation entities that support the particular connection. One such transfer syntax is discussed below. It corresponds to the specified by the OSI architecture for use together with the Abstract Syntax Notation One.

7.2.1 Tag Identifier

The identifier octets encode the tag class and number of the associated data type. The encoding of the identifier is given Table 7.1

If the tag number is between 0 and 30, one identifier octet is adequate so encode the tag. Additional octets are required to specify the tag number in case it is 31 or more. A data type whose tag is, for example, APPLICATION 180, the three identifier octets would be 01111111 10000001 00110100, where bit 8 of the second and third octets indicate whether at least one identifier octet follows the current octet, or not, and x is 0 or 1.

Table 7.1 Encoding of a Tag Class and Number: Leading Octet

Bits	Value	Interpretation
Bits 8 and 7	00	Universal class
	01	Application class
	10	Context-specific
	11	Private class
Bit 6	0	Primitive data type
	1	Constructed data type
Bits 5 through 1	11111	Tag number is 31 or more
	xxxxx	Binary representation of tag number

Note: Bit 8 is most significant bit, and bit 1 is the last significant.

Table 7.2 Encoding of Tags: an Example

Data type	Tag	Identifier octets
Employee Record	APPLICATION 0	01100000
Job Title Type	UNIVERSAL 22	00010110
(type of) Job Title	CONTEXT 1	10100001

7.2.2 Length of Contents

The length octet specifies the number of octets in the contents field. There are three distinct cases:

1. The data type is primitive, in which case the length specifies the exact number of octets in the contents field,
2. The data type is constructed, but the encoding of data value is unavailable at the current time. Here the length field is encoded to suggest that end-of-contents octets are present and will be used to delimit the contents field,
3. The data type is constructed, and the encoding for contents field is available. Here the sending Presentation entity has the option to either specify the length explicitly, or delimit the contents field with end-of-contents octets.

If the number of octets in the contents field is less than or equal to 127, then one octet is adequate, otherwise 2 or more octets are required to specify the length. The encoding of leading length octet is given in Table 7.3. In Table 7.4 the examples illustrate the encoding of the length field.

Table 7.3 Encoding of the Length Field: Leading Octet

Bits	Value	Interpretation
Bits 8 through 1	0xxxxxxx	Number of octet is less than 128, binary encoded
	10000000	End-of-contents octet is present
	11111111	Not used
	1xxxxxxx	Number of subsequent octets in the length field, binary encoded

Table 7.4 Encoding of the Length Field: Some Examples

Length	Encoding
18	00010010
180	10000001
1048	10000010 00000100 00011000
unknown	10000000

The end-of-contents octets, if present, are encoded as two zero octets. The question of ensuring transparency would be discussed shortly.

7.2.3 Encoding of Octets Field

The contents filed, consisting of zero or more octets, is used to encode the actual data value. This encoding depends upon its data type. If the type being encoded is primary, then it directly encodes the data value. Otherwise, if the type is constructed, then each component is encoded in a manner similar to the being discussed, that is, recursively using the 4-tuple (identifier, length, contents, end-of-contents).

The OSI standards specify the encoding of the contents field for each of the primary data types, as well as for those that are constructed. Below we discuss the encoding of some of these.

1. A value of type BOOLEAN is encoded as (00)H if the value is FALSE, and as a non-zero octet, otherwise.
2. An integer is encoded as one or more octets using Two's complement binary representation using a minimum number of octets. For example, - 25 is encoded as 11100111, or (E7)H, but not as (FF E7)H.
3. Consider now the encoding of a value of type OCTETSTRING. If the entire string of octets is available, then it may be treated as one long string of octets of whose length is known and indicated in the length field. The encoding of the octets is straightforward. But, when it is necessary to encode one part of the string at a time, then the string is viewed as constructed. The end-of-contents indicator is then used to delimit the construction. Each sub-string is transferred as a component with its own identifier, length, contents, and possibly end-of-contents octets.

4. A similar scheme may be used to encode a BITSTRING. Here, additionally, one needs to indicate the number of significant bits in the last octet. All data transfers, however, are of an integral number of octets.
5. A character string of the type ISO646STRING is encoded very much like an octet string, but after mapping each character onto an octet using the IA5 7-bit character encoding scheme. Bit 8 is set to zero.
6. A value of a data type that has been tagged is treated as constructed. As such the contents field consists of encoding of the base encoding.
7. As a last illustration of encoding of the contents field, consider the construction of the type SET. Its encoding is constructed. The contents field then consists of the complete encoding of each component of the data value. The detailed example below illustrates this.

7.3 Presentation Services

In this section we discuss services that the Presentation layer makes available. This enables Application entities to transfer information using a syntax of their choice. The applicable collection of abstract syntax may be agreed to at the time of connection establishment, and perhaps changed subsequently. Service primitives, and their parameters, are discussed in this section.

7.3.1 Context Establishment

The Presentation layer provides services using which Application entities may initially define the set of Presentation contexts, and subsequently modify it. The defined context set, to be initially applicable, is negotiated during connection establishment. The interface with the PS provider enables the initiating user to identify one or more abstract syntax to be supported by the PS provider. The connection establishment procedure is successful provided:

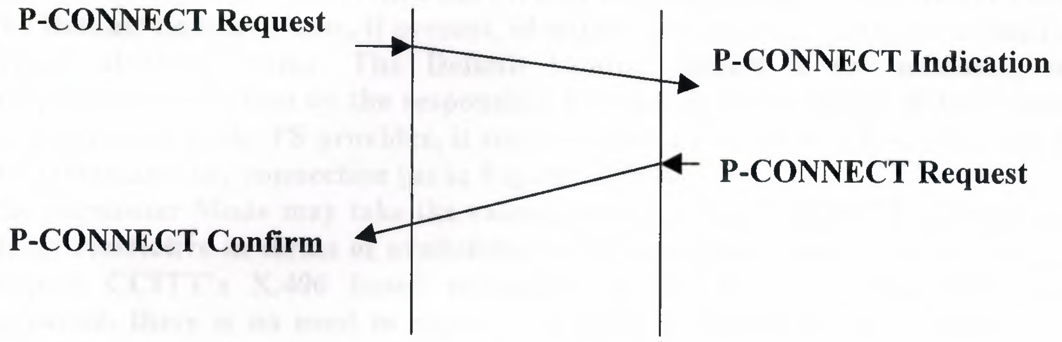
1. The PS provider is capable of supporting the default context and some, if not all, of the named abstract syntax using an appropriate transfer syntax.
2. The responding service user at the other end agrees to the proposed default context and, partly or wholly, to the proposed defined context set,
3. Together, the Presentation layer and the supporting Session layer are able to provide the required services.

If either of these conditions is not satisfied, then the establishment of the connection is unsuccessful. The three possibilities, indicated in the Figure 7.3, corresponds to:

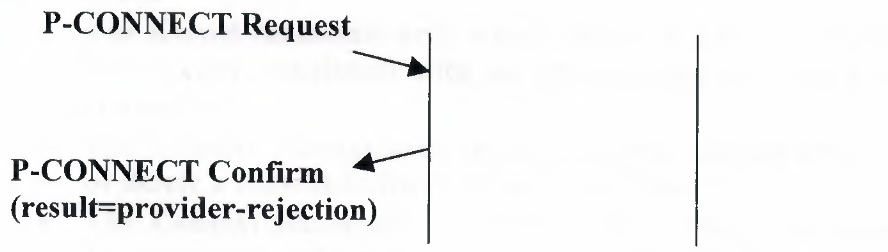
1. successful Presentation connection establishment,
2. connection establishment refused by the PS provider,
3. connection establishment refused by the responding PS user.

The P-CONNECT service is confirmed. We briefly discuss each of these parameters.

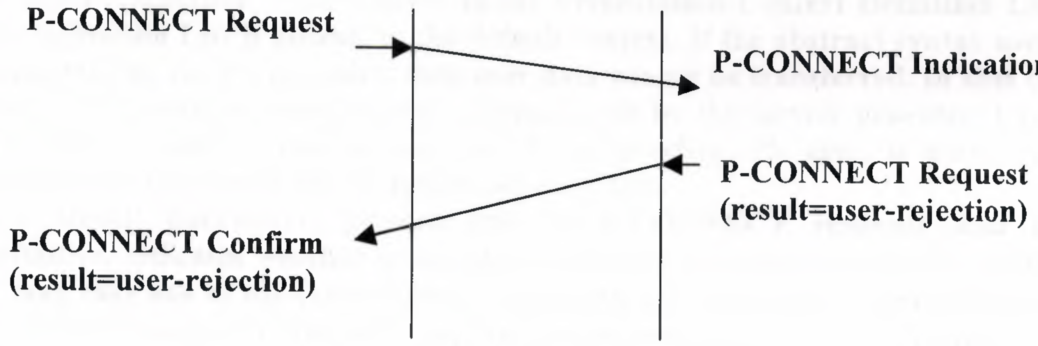
1. **The Calling and the Called Presentation Addresses are as usual PSAP addresses. The Called Presentation Addresses is mapped by the Presentation layer, using the address mapping function, to obtain a Called Session Addresses of a SSAP to which the supporting Session connection must be established. The Responding Address is present in the response and confirm primitives only if a PSAP address other than the Called Presentation Address should be used to re-establish a Presentation connection. This is helpful in case of generic addressing or in redirection.**



(a) Successful Establishment.



(b) Establishment rejected by the PS provider.



(c) Establishment rejected by PS user.

Figure 7.3 Establishment of a Presentation connection.

2. Whenever present, Presentation Context Definition List contains one or more items. Each item consists of two components, a Presentation context identifier and an abstract syntax name. The Presentation context identifier has significance only for the local PS user and the PS provider. It is assigned by the initiating PS user and interpreted by the PS provider. In all future references to a Presentation context only the context identifier is used.
3. The Presentation Context Definition Result List is again a list of one or more items, each of which indicates acceptance or rejection of the corresponding context in the proposed Presentation Context Definition List. Each value represents either acceptance, user-rejection or provider-rejection. Further, if a proposed context is rejected by the PS provider, then the PS user may not accept the particular context.
4. The Default Context Name, if present, identifies the abstract syntax to be used as the default abstract syntax. The Default Context Result is an indication of the acceptance or rejection by the responding PS user. If the proposed default context is not acceptable to the PS provider, it simply issues a P-CONNECT confirm primitive, and terminates the connection (as in Figure 7.3(b)).
5. The parameter Mode may take the value normal or X.410 (1984). The latter mode is highly restrictive in terms of availability of Presentation context, but is adequate to support CCITT's X.400 based messaging system through. With this mode of operation, there is no need to negotiate a defined context set or a default context. User data concerning this application is assumed to be of the type OCTET STRING.
6. Through Presentation Requirements, a user may specify the list of optional functional units of the Presentation service that it requires. Three functional units are defined:
    - The Kernel functional unit, which always available. It permits transfer of user information consistent with an abstract syntax from the prevailing defined context set.
    - The Context Management functional unit, additionally enables a user to add or delete a context to/from the defined context set.
    - The Context Restoration functional unit, when used together with Context Management, enables a user to request that the defined context set be restored to a defined context set prevailing at an earlier time. Restoration is carried out as part of session resynchronisation and activity management.
7. Information exchanged as User Data in P-CONNECT primitives may be expressed in any Presentation context listed in the Presentation Context Definition List or, if the Definition List is absent, in the default context. If the abstract syntax used is not supported by the PS provider, then user data cannot be transferred. In that case, the connection establishment attempt is terminated by the service provider. Further, if the context used is unacceptable to the responding PS user, it may reject the connection. Or, it may simply ignore the user data.
8. The Result parameter, present only in P-CONNECT response and confirm primitive, indicates whether or not the connection has been successfully established. It may take one of the three values, acceptable, user-rejection or provider-rejection. It, thereby, suggests to the initiating Application entity the cause of rejection.
9. The parameters, including Session Requirements, Quality of Service, Initial Synchronisation Point Number, Initial Assignment of Tokens, and Session Connection Identifier, are directly mapped onto parameters supplied within corresponding S-CONNECT primitives. Recall, a Presentation connection is supported by an underlying Session connection.

7.3.2 Other Presentation Services

Data transfer primitives. While user information may be transferred as part of almost any Presentation service primitive, the following service elements specifically support transfer of user information:

1. P-DATA,
2. P-TYPED DATA,
3. P-EXPEDITED DATA,
4. P-CAPABILITY DATA.

These are similar to the corresponding Session service elements. The difference, however, is that user information is encoded using an applicable context from the defined context set. Since segmentation is not permitted, the size of a Presentation service data unit is determined solely by the permissible Session SDU size. Needless to say, these Presentation services are available subject to the provision of corresponding Session services. In particular, P-DATA service is available with or without token control depending upon whether data transfer over a Session connection is half duplex or not.

Connection release. A Presentation connection may be terminated by its PS users or by the PS provider abruptly, or in an orderly manner. The corresponding Presentation service primitives are:

1. P-U-ABORT,
2. P-P-ABORT,
3. P-RELEASE.

Primitives concerning P-RELEASE service are mapped directly onto the corresponding Session service primitives. As such, rules governing P-RELEASE primitives are the same as those concerning S-RELEASE primitives. Further, the Presentation connection is released simultaneously with release of the supporting Session connection.

Synchronisation and token management. The remaining Presentation services are also derived directly from those made available by the Session service. This allows a PS user to effectively access these Session services. The PS provider, on its part, simply passes the semantics of the primitives onto corresponding Session primitives. The Presentation services that fall in this category include:

1. P-TOKEN GIVE, P-TOKEN-PLEASE, and P-CONTROL GIVE,
2. P-SYNC MINOR and P-SYNC MAJOR,
3. P-U-EXCEPTION REPORT and P-P-EXCEPTION REPORT.

Further, since most of the Presentation service primitives are directly (or indirectly) mapped onto corresponding Session service primitives, these are more or less identical to those specified by the Session service and its protocol. We shall, therefore, limit the discussion to P-ALTER CONTEXT primitives for which there are no corresponding constraints implied by the Session protocol.

7.4 Presentation Protocol

The Presentation protocol specifies the functions required to be implemented by the Presentation layer in order to close the gap between Presentation services and those available by the Session layer. A quick comparison between the two services immediately reveals that these functions are basically related to negotiation of the use of one or more Presentation context, and to encoding of user information in an appropriate transfer syntax.

7.4.1 Connection Establishment

The connection establishment procedure is used to establish a connection between two communicating Presentation entities, and as a consequence, between the supported PS users. The procedure specifies the use of CP, CPA, and CPR protocol data units. Figure 7.4 illustrates the resulting three different possibilities, corresponding to

1. successful connection establishment,
2. connection is rejected by the responding Presentation entity,
3. connection is rejected by the responding PS user.

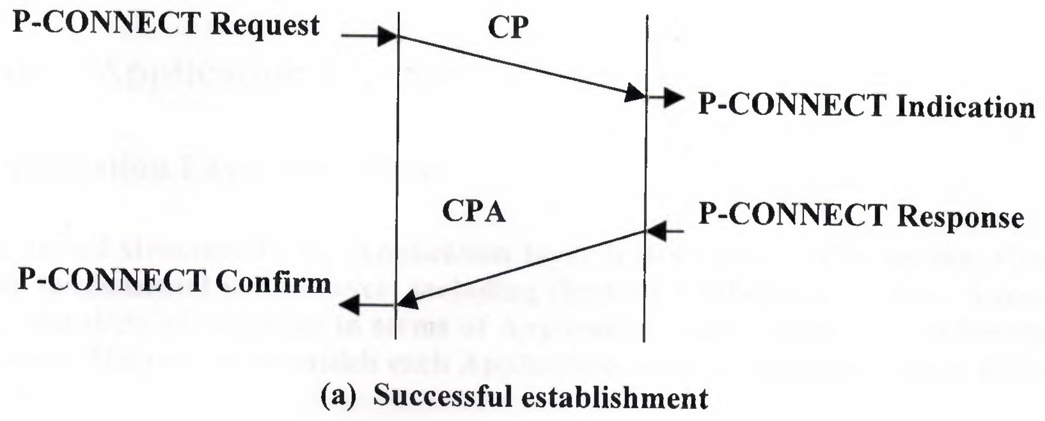
A connection is successfully established provided negotiation is respect of each of the following is successful :

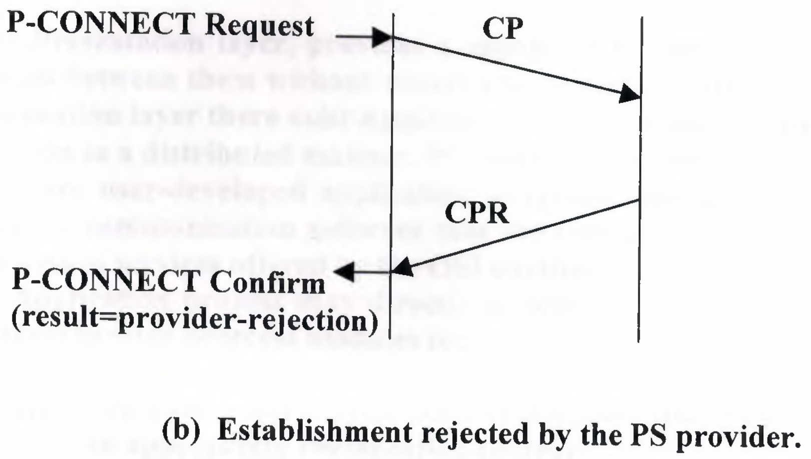
1. defined context set,
2. default context,
3. Presentation functional units and Session services,
4. version of the Presentation protocol.

The negotiation generally take place between the four entities, the two PS users and the two supporting and the two supporting Presentation entities. But, clearly the version of the Presentation protocol is concern only to the Presentation entities. Further, use and availability of Session services is determined by the Session layer as well. The negotiation procedure for each of these is necessarily different.

The negotiation procedure for the defined context set is described below.

1. For each abstract syntax requested by its PS user, the initiating Presentation entity indicates to its peer entity, in a CP PPDU list of transfer syntax's it is capable of supporting.
2. The responding Presentation entity indicates, in the indication primitive it issues to the corresponding PS user, those abstract syntax's that it can (or cannot) support.
3. The responding PS user indicates to its supporting Presentation entity those abstract syntax's that it can use. This it does by issuing a P-CONNECT response primitive.
4. The responding Presentation entity sends out a CPA or a CPR PPDU indicating, therein, the selection of a transfer syntax for each accepted abstract syntax. For each abstract syntax not accepted, it conveys the source of rejection. A reason is provided if the abstract syntax is rejected by the responding Presentation entity itself.




P-CONNECT Request
CP
CPR
P-CONNECT Confirm
(result=provider-rejection)
(b) Establishment rejected by the PS provider.

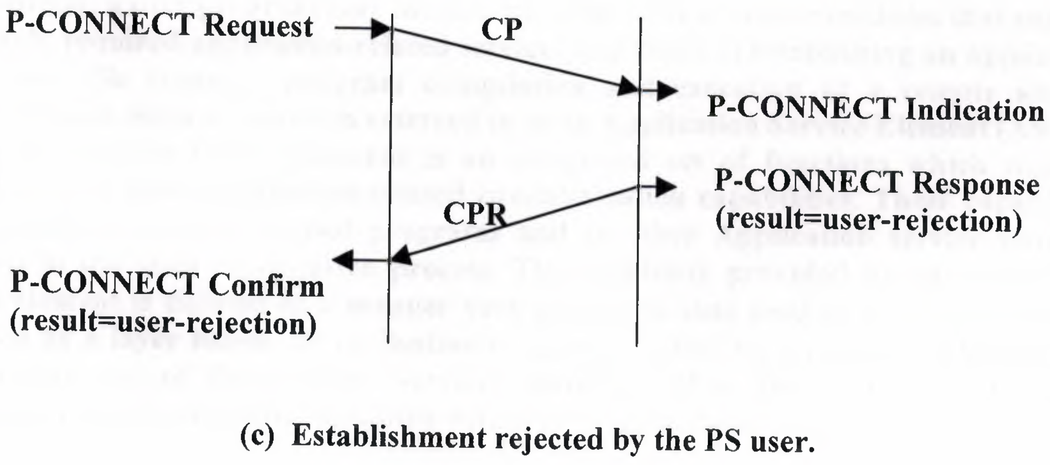


Figure 7.4 Exchange of CP, CPA, CPR PPDUs in connection establishment.

CHAPTER 8

Common Application Services

8.1 Application Layer Structure

The broad structure of the Application layer is described in this section. Concepts that are fundamental to this layer, including those of Application process, Application entities and their composition in terms of Application service elements are discussed in this section. The manner in which each Application entity is described is also covered in some detail.

8.1.1 Application Processes

The Presentation layer, provides a capability to users so that information may be exchanged between them without concern for its representation, or its transfer. Within the Application layer there exist Application Processes that use this capability to process information in a distributed manner. Physically, an Application process is a collection of one or more user-developed application programs and communication software. It is through the communication software that the ultimate user or an application program gains access to services offered by the OSI environment.

An Application process may directly interface with the Presentation layer. In that case it must include protocol modules to:

1. **initialise communication with its peer Application processes,**
2. **establish an appropriate Presentation context,**
3. **transfer files or messages as necessary, etc.**

Alternatively, a user program may include an instance of available modules that support commonly required application-related services like those of establishing an application association, file transfer, program compilation and execution at a remote site, or electronic mail. Such a module is referred to as an Application Service Element (ASE).

An Application service element is an integrated set of functions which together provide one or more application-related communication capabilities. These capabilities are available to user-developed programs and to other Application service elements included in the same Application process. The capability provided by an Application service element is defined in a manner very similar to that used to specify the service provided by a layer below. Its realisation is again specified by a protocol. The protocol may specify use of Presentation services directly and/or those provided by other Application service elements contained within the Application process.

8.1.2 Application Entities

Open Systems Interconnection is primarily concerned with aspects that relate to interaction between Application processes residing in possibly different systems. The notion of an Application entity (AE) is, therefore, defined. An Application entity is a

model of those aspects of an Application processes. Such a model views a user program, together with its interface with Application service elements that is uses, as a User Element. An Application entity is, therefore, composed of a collection of one or more Application service elements that a user element accesses to perform distributed processing of information.

A user element is a model of the particular service that an Application process makes available to, for instance, human users, other programs, or devices that are outside the OSI environment. As such, the OSI architecture does not address issues relating to the design of the user element, except the insist that its interface with the Presentation layer conforms to the standards.

8.1.3 Application Association

Just an Application process needs to be activated before information processing can begin, so must an Application entity be invoked. Each invocation of the Application entity represents the use of some capability provided by one of its service elements. Necessarily then, an invocation results in some form of interaction between the Application entity and its peer entity. There must, therefore, exist an association between the pair of Application entities. Such an association is called Application association.

An Application association between two Application entities is a relationship between them which not only establishes the common communication environment initially, but provides a basis for co-ordinating interactions between them. The environment, for instance, defines what capabilities are available in terms of Application service elements, and the nature of user data be communicated using these capabilities. Thus, the aspects that are negotiated at the time of association establishment are the :

The Application Context, and the Abstract Syntax.

The Application Context is the common environment shared by the two communicating Application entities. It essentially comprises of the list of Application service elements, and the particular capabilities that each service element provides. The latter is particularly significant if there are optional facilities provided by an Application service element, or if the required protocol permits a choice in using certain procedures.

The Abstract Syntax is a specification of the syntax of Application PDUs that are communicated over a supporting Presentation connection. It consists of the abstract syntax of PDUs that relate to the various Application service elements that comprise the Application entity, and of information communicated between the user elements.

8.1.4 Application Service Elements

As a matter of judgement, Application service elements (ASEs) are subdivided into two groups:

1. common Application service elements (CASE),
2. specific Application service elements (SASE).

The common ASEs provide services that may be used by a user to element, a common ASE or a specific ASE. Four common ASEs are currently defined. These are:

1. Association Control service element (ACSE), which enables a user to establish or terminate an Association between Application entities,
2. Reliable Transfer service element (RSTE), which enables reliable transfer of information between peer entities.
3. Remote Operations service element (ROSE), which permits users to initiate operations at a remote site,
4. Commitment, Concurrency and Recovery services (CCR), which enable users to recover from a failure during execution of a task using commit or rollback procedures.

The common Application service elements are discussed in the remaining sections of this chapter. Other Application service elements also provide services that may be used by a variety of Application service elements as well as by a user element. These services are not as generic, and are, therefore, useful only in specific instances.

8.2 Association Control Services

As discussed in the previous section, a common Application service element (ASE) is a collection of modules, which provides a service to a user element or to other ASEs of the same Application entity. The Association Control service element (ACSE) is discussed in this section.

The concept of an Application Association is central to the discussion of Application layer services. Its main purpose is to establish the correct environment for information exchange, negotiation of the Application context, the set of required Presentation contexts, and the set of required Presentation layer functional units. The Application context is a specification of the set of Application entities, together with related options. When used, the Association Control service element is always included in the Application context.

The negotiated Presentation context set includes the abstract syntax required to convey protocol data units (PDUs) of all Application service elements specified in the Application context, including ACSE, and the user element. Such PDUs are called Application PDUs or simply APDUs.

Yet another purpose of establishing association is to properly identify the source and destination Application entities. Application entities are referred to by names. An Application entity title (AE title) is composed of an Application process title(AP title) and an EA qualifier. The latter uniquely identifies the particular Application entity which models only a part of the Application process. Further, since there may be multiple invocations of the same process, it is necessary to distinguish one invocation from the others. An Application entity is, therefore, identified by:

1. AP Title,
2. AE Qualifier,
3. AP-Invocation-Identifier,
4. AE-Invocation-Identifier.

8.2.1 ACSE Services

Services included in ACSE are those of:

1. **association establishment,**
2. **orderly or negotiated termination of the association, which ensures that there will be no loss of information in transit,**
3. **user-initiated or provider-initiated abrupt termination of the association, which may result in loss of information during transit.**

Negotiated termination permits a responding user to either accept or reject a termination of the association. Orderly release, on the other hand is not subject to availability of release token. The correspondence between the above service elements and ACSE service primitives is given in Table 8.1. With termination of the association, the negotiated Application and Presentation contexts are no longer valid. There must be re-negotiated if the association is re-established.

Table 8.1 Association Control Service Primitives

Service	Service Primitives	Type of Service
Association Establishment	A-ASOCIATE	Confirmed
Normal Release	A-RELEASE	Confirmed
Abnormal Release	A-ABORT	Unconfirmed
	A-P-ABORT	Provider-initiated

CHAPTER 9

Directory Services

9.1 Introduction

A Directory holds information on a collection of objects in the real world. The information is structured so as to permit Application processes to read, search or modify information concerning one objects. Such a capability to access information can be very useful in applications which require, for instance, name-to-address translation, or which simply seek related information on people or equipment. In particular, OSI applications need to determine the address of the Presentation service access point through which an Application entity can be reached, and the title of which is known. OSI network management protocols could also use a directory to access information on various network resources, including gateways, or other servers.

More formally, a directory is a collection of open sub-systems which co-operate between themselves to hold information about a variety of objects. The users of the directory include human users as well as computer programs. A program is an embodiment of another Application service element or of a user element of an Application entity. Users can read, search, or modify information concerning one or more named objects. From the OSI viewpoint, each user is represented as a Directory User Agent (DUA), while the directory is itself composed of one or more co-operating Directory Service Agents (DSAs). Permissible operations on a directory are modelled as directory services, and made available to DUAs at well-defined service access points (see Figure 9.1).

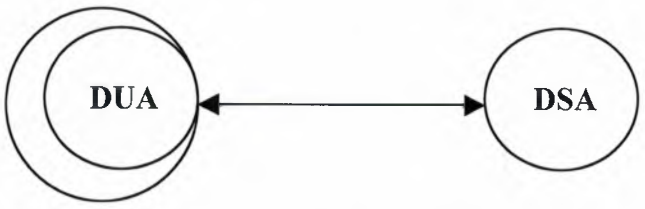


Figure 9.1 Model of directory user and service agents.

9.2 Directory Information Base

For the present, we assume that the directory is not distributed. The information held in a directory is collectively known as a Directory Information Base (DIB). It is composed of a number of directory entries. Each entry stores information about an object, and is made up one or more attributes of the object. The collection and type of attributes stored in an entry is dependent upon the object class to which the object b[illegible]s. An object class is an identified collection of objects which share certain common characteristics. As such, each object class is described in terms of a common set [illegible]ributes. Some or all attributes may be optional.

9.3 Directory Information Tree

Given the above definition of the DIB and the structure for an entry, one may maintain a directory in the form of a flat file containing a list of entries, where each entry stores the set of classes to which the corresponding object belongs, and the required attributes. Since the number of entries can be very large, such a structure is not efficient, both from the viewpoint of storage and access. Fortunately, directory information naturally reflects a hierarchical relationship between objects. It is this hierarchy that is exploited in maintaining the directory as a tree.

Thus, a directory is organised in the form a tree, called Directory Information Tree (DIT). A Directory Information Tree is illustrated in Figure 9.2, where each vertex, other than the root, represents an entry corresponding to an object. It is excepted that vertices nearer the root of tree will represent objects belonging to large classes, countries or organisations for instance, whereas leaf vertices will represent people, equipment or Application entities.

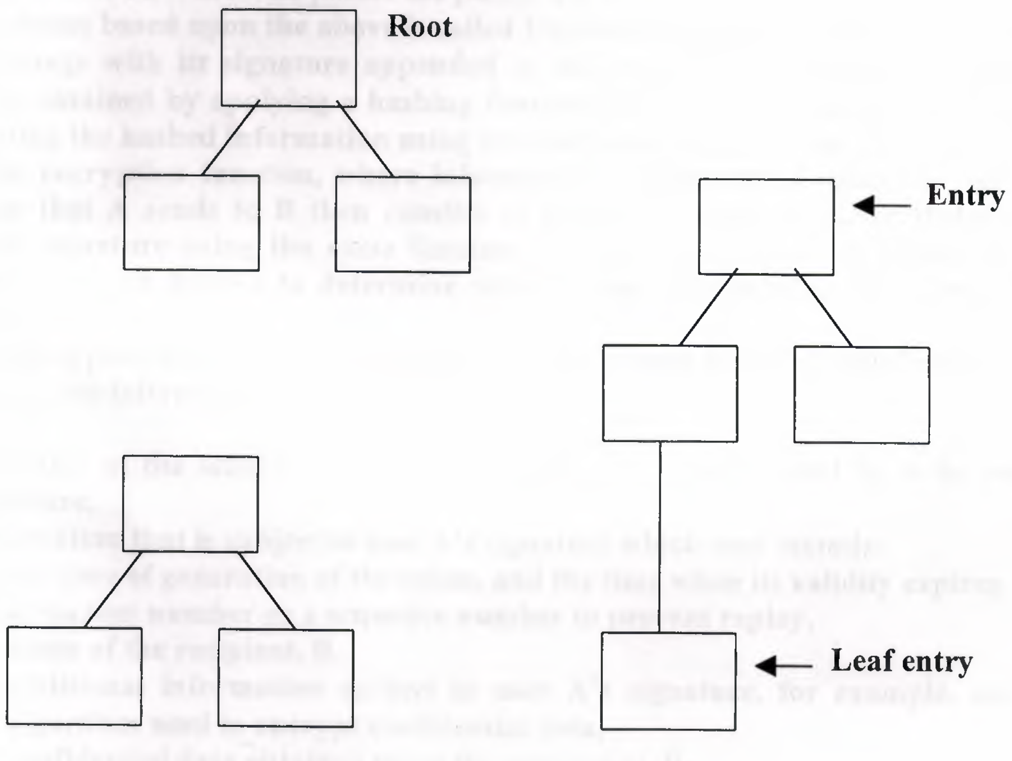


Figure 9.2 Structure of the Directory Information Tree.

9.4 Authentication

One of the uses to which the directory can be put is to authenticate a user's identify. Further, since the directory holds information on various objects, it must itself be protected from unauthorised access. The extent to which two users make their communication secure depends upon the application. This in turn determines the level of security one builds into the system. Surely, when a user accesses the directory, both the DSA as well as the DUA must be sure of each other's identity. (Authentication is also relevant when two DSAs communicate with each other.)

The approaches are purposed in [CCITT X.509], both of which provide different levels of authentication of requester's identity. The second approach also ensures that requests can only be interpreted by the intended recipient. The first approach, also called simple authentication, requires a user to furnish not only its distinguished name, but also a password which may be checked against a value stored in the directory. The stored value is an encrypted form of user's password, to prevent users from reading it. This scheme, obviously, provides limited security.

9.4.1 Public Key Encryption

The second approach, also called strong authentication, is based upon the use of public key encryption. It involves the use of a pair of distinct but complementary keys, a secret key and a public key. The identify of a user can be authenticated, if it can be determined that user A possess its secret key, A_s. For another user to authenticate the identity of user A, it must be posses the public key of user A, A_p.

A scheme based upon the above is called Digital Signature. Basically, user A sends to B a message with its signature appended to the message. The signature consists of a message obtained by applying a hashing function, h, to the information, info, and then encrypting the hashed information using its secret key, A_s, to obtain signature = X. Here X is the encryption function, where information I is encrypted using the key, k. The message that A sends to B then consists of [A, info, signature]. User B decrypts the received signature using the same function X, using the key A_p to obtain X. It then compares it with h(info) to determine whether the user sending the message is one claimed.

An encrypted message, sent by user A to another user B, is also called token. A token consists of the following:

1. the name of the sender, A, and the encryption algorithm used by A to obtain the signature,
2. information that is subject to user A's signature which may include:
   (a) the time of generation of the token, and the time when its validity expires,
   (b) a random number or a sequence number to prevent replay,
   (c) name of the recipient, B,
   (d) additional information subject to user A's signature, for example, encryption algorithm used to encrypt confidential data,
   (e) confidential data obtained using the public-key, B_p;
3. user A's signature, obtained by encrypting a hashed version of the above information using an encryption using an encryption algorithm mentioned in 1. Above, and A's secret key, A_s.

If confidential data is included, such data can only be interrupted by the intended recipient, B. The recipient uses its secret key, B_s to decrypt the confidential data, for instance, may be encryption key itself, used to encrypt data in subsequent data transfers.

9.5 Directory Services

9.5.1 Directory Operations

A directory provides three different sets of operations. Read-access, search-access, and modify-access. These correspond to three different types of ports, and may be used to limit a user's access to only those operations that are permitted by its access rights. As a consequence, a DUA may only interact with another DSA. If two DSAs are to interact with each other, they may not do so through any of these ports. Further, the operations may only be invoked by the DUA, whereas a DSA simply responds to user request for operations. These operations and their required arguments are discussed below:

1. The *read* operation is used to obtain information concerning a named entry. More specifically, the operation causes the values of some or all attributes to be returned. The DUA is required to identify the attributes of interest.
2. A *compare* request is similarly aimed at a particular entry in the DIT, and can be used to verify whether an attribute value supplied by the user matches the one stored in the entry. As an example, this facility can be used to check the user's own password. The attribute value itself may not be accessible for a read operation. Even if it can be read, it may not be interpretable since the stored value may be in an encrypted form.
3. The operation, *list*, causes the directory to return a list of immediate child entries of a named DIT entry. Specially, the DIT returns the RDN of each child entry.
4. The operation, *search*, as it name implies, permits a user to obtain information on several entities within a sub-tree of the DIT. Specifically, the attribute values of only those entries are returned which satisfy a certain property. The property is specified by the user in the form of a filter, discussed below.
5. An *abandon* request can only be made when one of the above operations is outstanding. It causes the directory to stop processing an earlier request and to provide all available results. The DSA is, of course, free to discard the results so far available to it.
6. An *add entry* operation permits the user, with appropriate access rights, to add a leaf entry of named entry. The entry to be added may either be an alias entry or it may correspond to a real object. A repeated use of this service will permit addition of an entire sub-tree to the DIT.
7. The *remove entry* operation causes the directory to delete the named leaf entry from the DIT.
8. One or more attribute values in a named entry can be changed using the *modify entry* operation. Attributes can also be added or deleted. Further, one or more values of an attribute may be removed, added or changed. It may also be used to change an alias entry, thereby referencing a different object entry in the DIT. Before carrying out any change, the directory ensures that the resulting tree remains consistent with the scheme used to design the DIT. That is, the resulting attribute types must be consistent with the object class, and their values with the attribute data types.
9. The operation *modify RDN* causes the RDN of a named leaf entry to be changed. A different RDN is provided by the user in the form of changes to the list of attribute types that determine the RDN, and/or their distinguish values. It does imply that part of the entry information may also change, as a consequence.

9.5.2 Parameters

A directory operation is specified in terms of:

1. **the data type of the argument to be conveyed as part of the remote operation request to the DSA (Directory Service Agent),**
2. **the data type of the result to be conveyed to the DUA which initiated the particular directory access, in case the remote operation is performed successfully,**
3. **a list of possible error conditions that may be encountered.**

The parameters associated with the read operation, as well as others, are listed in Table 9.1 .

Table 9.1 Argument and Result Parameters of Directory Operations

Operation	Argument parameters	Result parameters
read	object-name entry-information-selection common-arguments	entry-information common-results
compare	object-name attribute-value-assertion common-arguments	matched common-results object name
abandon	invoke-ID	-
list	object-name common-arguments	object-name set-of-RDNs
search	base-object-name filter entry-information-selection common arguments	object-name set of entry-information
add entry	object-name set of attributes common arguments	-
remove entry	object-name common-arguments	-
modify entry	object-name set of entry-modifications common-arguments	-
modify RDN	object-name RDN-changes common-arguments	-

The parameter *common-arguments* is a set of parameters commonly used with a number of directory operations. It optionally includes:

1. *service-controls*, which may be used to indicate
   (a) priority level for the operation,
   (b) a time limit during which the operation must complete,
   (c) a maximum number of entries to be searched or listed,
   (d) the scope of the search,
   (e) whether chaining is allowed, preferred, or prohibited;

2. *security parameters* ensure that communication between the requesting DUA and the DSA is secure. It includes information which may be used to verify the user's identify, the identify of the DSA, and integrity of argument parameters;
3. the distinguish name of the requesting DUA.

The parameter, *invoke-ID*, of the abandon operation, is an identifier associated with an earlier operation which is to be abandoned. This parameter is assigned by the directory protocol at the time of invoking a remote operation, and made available to the directory user, for later reference.

The parameter, *filter*, is a logical expression of one or more attribute value assertions. It is used in a search operation to specify the entries that are of interest, and whose attributes are sought. The search space is relative to the entry whose *base-object-name* is supplied in a search operation.

The parameter, *set of attributes*, of the add-entry operation is the information to be stored in the entry to be added. The *set of entry-modifications*, used in a modify entry operation, is a selection of attributes, together with any relevant values, and the nature of modifications.

The parameter *common-results* includes the security parameters discussed earlier, and whether the entry that was finally referenced is an alias entry or the one pointed to by an alias. *Entry-information* is the information returned by the directory in response to a read or search operation. The *matched* parameter with a value true is returned if the corresponding attribute-value-assertion, in a compare operation, evaluates to true.

9.6 Directory Protocols

There are two protocols, Directory Access Protocol to support access to directory services, and Directory Systems Protocol which supports abstract operations at chained access points.

A DUA and a DSA are each Application processes, but modelled as Application entities from the viewpoint of the OSI architecture. The Application service elements specifically supported by the directory protocol are read-access, search-access, and modify-access. Thus any Application process which uses directory operation must include these service elements as well as ROSE and ACSE service elements. These service elements are used by the directory implemented Directory-Bind, Directory-Unbind, and directory access operations.

The directory access protocol uses the ACSE services to support Directory-Bind and Directory-Unbind operations. The operation, Directory-Bind, is directly mapped onto the A-ASSOCIATE primitives, with the following correspondence between the parameters of A-ASSOCIATE primitives and the Directory-Bind operation:

1. The Application Context Includes ACSE, ROSE, Directory-Bind, Directory-Unbind, read-access, search-access, and modify-access.
2. The Presentation context definition list includes the abstract syntax associated with ROSE and ACSE protocol and the abstract syntax of attributes stored in various entries, and argument/result parameters of directory operations.
3. The arguments of the Directory-bind operation are mapped onto the user-data parameter of A-ASSOCIATE request/indication primitives, whereas result or error parameters, depending upon whether or not the bind operation is successful, are mapped onto the user-data field of A-ASSOCIATE response/confirm primitives.
4. The directory access protocol requires only the Kernel and Duplex functional units of the Session service.

The Directory-Unbind operation is mapped onto the A-RELEASE primitives. It is expected that the release operation will always succeed.

CHAPTER 10

Message Handling System

10.1 Introduction

Electronic messaging, is an important application of computer networks. In the context of OSI, this application supported by a Message Handling System (MHS). The MHS enables users to exchange messages on a store-and-forward basis (see Figure 10.1). It also permits messages to be stored within the MHS till such time a user wishes to retrieve his/her messages. The MHS system, described in a series of documents, referred to by X.400, caters to a wide variety of applications, some of which are discussed below.

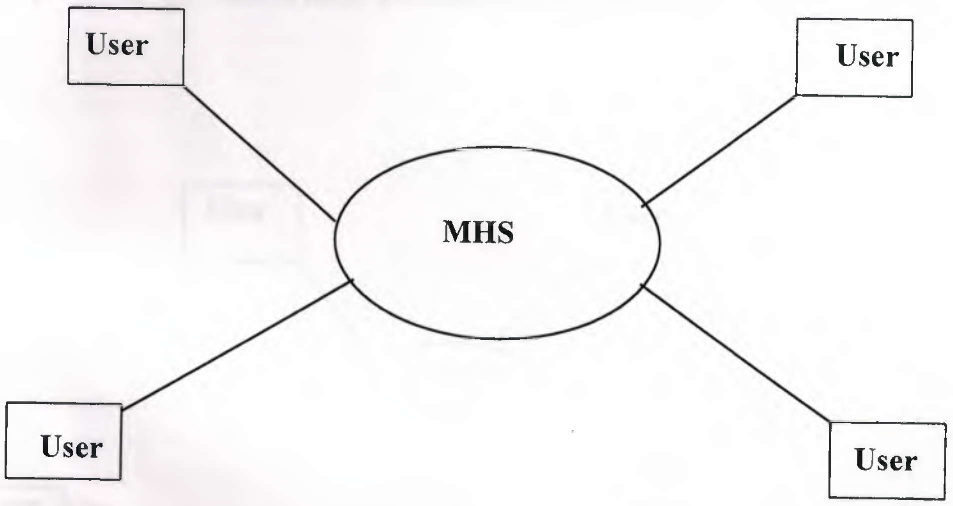


Figure 10.1 The Message Handling System and its users.

A message may be simultaneously sent for delivery to a number of users who have direct access to the MHS. The user originating the message either provides the names (or addresses) of individual users, or a list of user names in the form of a distribution list. The MHS architecture also provides sending messages through the network for delivery to users who do not have direct access to the MHS, but using other services including Telex, FAX and postal services.

The message content is not limited to a string of bits, bytes or charters. It may even be a voice, video or FAX message, but suitably encoded for message transfer and delivery. Such messages are called *multi-media-messages*. In either case, the message is transparently delivered by the MHS system to the intended user, except when the user originating the message requests format conversion to take into account differences in terminal devices. The need for format conversion can, instead, be determined by the MHS based upon its knowledge of the recipient's terminal capabilities.

There is also a provision in MHS for a user to determine, *a priori*, whether it is possible for the MHS to convey a message to a named user, or not. Such a user request is called *probe*. Depending upon the application, a user may request that the delivery of a message be confirmed by the MHS. The response of the MHS system to a message transfer or probe request is contained in a *delivery report.*

10.2 MHS Architecture

A model of the MHS, together with its major components is shown in Figure 10.2. A message submitted by a user, through its User Agent, is transferred transparently through Message Transfer System to the named user. The recipient is also connected to the MHS through its own User Agent, or through other means. A User Agent (UA) is an Application process in an open system which enables a user to access MHS capabilities, including submission of messages and reports. From the OSI architecture viewpoint, the user is modelled as the User element of the Application entity, and is thus identified by the corresponding UA. There is, as a consequence, exactly one UA Per user.

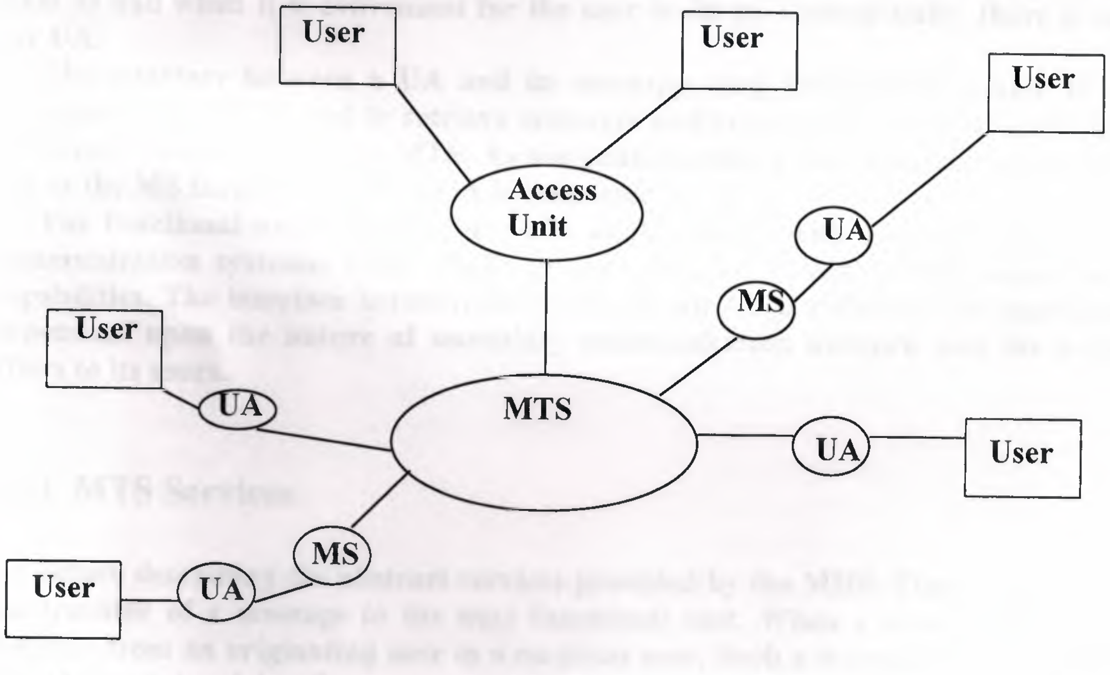


Figure 10.2 The Message Handling System and its components.

10.2.1 Message Transfer System

The Message Transfer System (MTS) is a collection of Application processes collectively responsible for conveying messages between UAs, as well as probes and reports between UAs and the MTS. The MTS is a store-and-forward communication network consisting one or more Message Transfer Agents. Each Message Transfer Agent (MTA) is capable of receiving messages from Uas or from other MTAs, and storing them. It may subsequently deliver the stored message to connected UA if the message is destined to the corresponding user, or forward the message to another MTA depending upon the route selected. Within an MTS there may be any number of MTAs, and each MTA may be connected to none, one or more Uas and MTAs.

10.2.2 Message Store

An MTA performs a number of other functions, including routing, authentication checks, message conversion, etc. But the one function it does not perform is to store messages so that users may retrieve messages and process them it their convenience. A Message Store is an Application process which provides an alternative method to interface a UA with the MTS. Aside from enabling users to submit messages and probes to the MTS, the MS takes delivery of messages on behalf of the user from the MTS. Further, it permits the user to selectively retrieve messages, store them, and process them as and when it is convenient for the user to do so. Conceptually, there is one MS Per UA.

The interface between a UA and its corresponding MS enables a user to submit messages to the MTS, and to retrieve messages and reports. The services provided to a UA directly connected to the MTS. As one consequence, a user may be identified by the UA or the MS through which its UA is connected.

The functional unit, Access Unit (AU), enables users with access to other forms of communication systems, Telex, FAX, or even postal services, to gain access to MHS capabilities. The interface between the MTS and an AU is different. Its specification is dependent upon the nature of secondary communication network and the services it offers to its users.

10.3 MTS Services

Before describing the abstract services provided by the MHS. Figure 10.3 illustrates the transfer of a message to the next functional unit. When a user wishes to send a message from an originating user to a *recipient* user. Such a transfer requires a number of steps, each involving the movement of the message to the next functional unit. When a user wishes to send a message to its peer, it makes the message available to its UA. This step is called message origination. Subsequently, the UA *submits* the message to an MTA within the MTS. It is the responsibility of the MTS to move the message across the MTS, by *transferring* it from one MTA to the next MTA, depending upon the selected route. Ultimately, the MTS *delivers* the message to a UA. A user is then said to have *received* the message. In case an originating user (or UA) accesses the MTS capability through an MS, submission of a message by the UA to the MS is said to be *indirect submission.* It is the MS that finally submits to message to the MTS. Further, if a recipient UA takes delivery through an MS, the message may be *retrieved* at any time. If a message is submitted though an Access Unit which supports non-MHS systems, the message is said to have been *imported.*

10.4 MTS Operations

An MHS operation is specified in terms of:

1. the data type of the argument to be conveyed as part of the operation request,
2. the data type of the result to be returned, in case the MHS operation is performed successfully,
3. a list of error conditions together with the type of information returned with each error.

10.4.1 Submit Operations

Message submission. The operation, Message submission, may be invoked by a user to request transfer and delivery of a message to one or more MTS-users. Successful completion of the operation only signifies that the message has been accepted by the MTS. A successful delivery can only be confirmed by a delivery report from the MTS. These are listed in Table 10.1.

Table 10.1 Argument and Result Parameters of Message Submission

Argument parameters	Result parameters	Errors
originator	submission identifier	submission control violated
recipient (s)	submission time	service not subscribed
priority	MTA certificate	originator invalid
conversion	proof of submission	recipient improperly specified
delivery time	content identifier	inconsistent request
delivery method		security error
physical delivery mode		unsupported function
report request		remote bind error
security		
contents		

1. The parameter, *Recipient*(s), consists of a list of intended recipients and distribution lists. The MTS may additionally specify an alternate recipient for each intended recipient. Further, an alternate recipient, itself, or by the MTS. The originating MTS-user has the option to allow or prohibit substitution of recipient names.
2. *Conversion*-related parameters allow a user to explicitly specify a conversion of the message format. Further, it may allow or disallow any implicit conversion is necessary for the message to be delivered.

3. *Delivery time* arguments allow a user to request the MTS to defer delivery of a message to a specified time, or to place a limit on the time before which the message must be delivered.
4. Since a message may be delivered using non-MHS media, the user has the option to specify the physical delivery mode to be used to deliver the message, or to prohibit physical delivery.
5. Report *request* parameter is used to request a delivery report and to specify its nature.
6. *Security* related parameters include the originator's certificate and a message token. The message token is used by the MTS to authenticate the origin of the message. It uses the originator's certificate to obtain a verified copy of user's public key. Further, the message token can also be used to send confidential information, and whose integrity is not compromised. Using other security parameters, an MTS user may request a proof-of-submission and proof-of-delivery.
7. *Contents* parameters include *content identifier*, an indication of the content type, *the type of encoding*, and the contents themselves.

If the operation, message submission, is performed successfully, the MTS returns a result reply with the result parameters, listed in table 10.1.

1. Message *submission identifier* uniquely identifies the submission when a delivery report is given to the MTS-user, or by the MTS-user itself in a subsequent request to cancel its delivery.
2. Content identifier identifies the contents in the corresponding message submission request.
3. Message submission time is the time the message was submitted to the MTS.
4. Originating MTA certificate contains the MTA's public key, using which the originating MTS-user may authenticate the origin of the result reply. The originating MTA is the one with which the MTS-user interfaces, and is also the one that acts on behalf of the MTS.
5. Proof of submission parameter provides the MTS-user with a proof that the identified message was indeed submitted, and at the stated time. The proof is of the form, message submission identifier and time, together with the MTA's signature.

10.5 MTS Protocols

Above, we have discussed services and operations concerning three different pair of MHS objects, that is:

1. MTA and an MTS-user,
2. MS and an MS-user,
3. two MTAs.

As a consequence, three different protocols are required. These are:

1. MTS Access Protocol (P3),
2. MS Access Protocol (P7),
3. MTS Transfer Protocol (P1).

10.5.1 MTS Access Protocols

It assumed that Remote Operations Service Elements (ROSE) are available to support MTS services. They provide the request/reply paradigm required by the MTS operations available at different ports. Mapping functions are used to map MHS operations onto services provided by ROSE primitives. The service elements, submission, delivery, and administration, are, in fact, the functions that map operations of corresponding ports. Mapping functions are used to map MHS operations onto services provided by ROSE primitives. The service elements, submission, delivery, and administration, are in fact, the functions that map operations of corresponding ports.

REFERENCES

Open Systems Interconnection Revised Addition

Its Architecture & Protocol ....BİJENDRA N. JAİN ASHOK K. AGRAWALA

Data Communication Computer Network & Open System.....................FRED HALSAL