Chapter 3

OSI Upper Layer Architecture and Model: State of the Art

This chapter discusses aspects of the OSI-RM related to the development and definition of application protocols. The OSI-RM has been adopted by ISO and ITU-TSS as the major framework for coordinating protocol standardization efforts. In addition, the wide acceptance of the terminology defined in the OSI-RM has contributed to the establishment of a ‘common language’ for discussing protocol engineering concepts across different protocol architecture communities.

The OSI-RM comprises both a (loosely defined) design model and a reference architecture for protocol design. The design model defines such general concepts as service, service primitive, service data unit, service access point, protocol, protocol entity, and protocol data unit. It also defines the basic structuring technique for OSI: layering. Extensions to this design model were later defined by the OSI Application Layer Structure (OSI-ALS). The OSI-ALS is mainly concerned with additional structuring techniques that must enable a greater flexibility in combining Application Layer standards.

The reference architecture defines a layered subdivision of the overall protocol functionality. This results in a global characterization of four lower protocol layers and three upper protocol layers. The lower layers are concerned with data transfer and transmission functions, whereas the upper layers are concerned with distributed application interactions. We will refer to the design model aspects related to the upper protocol layers as the OSI Upper Layer Model (OSI-ULM) and to the definition of the three upper protocol layers as the OSI Upper Layer Architecture (OSI-ULA).

Requirements with respect to application protocols continuously evolve and therefore require the OSI-ULA to be open-ended. The structuring techniques of the OSI-ULM determine to what extent it is possible to support incremental extension of application protocol functionality and re-use of available application protocol standards.

The purpose of this chapter is to present the OSI-ULM and the OSI-ULA so that their description can be used as a reference for evaluation and comparison. The chapter explains the structuring techniques, and related concepts, that are adopted for application protocol development in the context of OSI. It also explains the subdivision of application protocol functionality according to the OSI-ULA.

The structure of this chapter is as follows: section 1 discusses structuring techniques that have been used for the composition of application protocol standards; section 2 provides an overview of the most important application service and protocol standards of the present OSI-ULA; section 3 presents more details on the Session Layer standards; section 4 presents more details on the Presentation Layer standards; section 5 presents two of the ‘building block’ standards of the Application Layer; section 6 presents more details on the Distributed Transaction Processing standard; and section 7 presents the conclusions of this chapter.

3.1 Application protocol structuring techniques

We identify three techniques that are available in the OSI-ULM for structuring application protocols:

- *layer composition* (or layering): This is the basic structuring technique in OSI that effects a separation of concerns through a subdivision of the overall protocol functionality into hierarchically related protocol layers. A protocol layer consists of protocol entities that interact peer to peer using the service provided by the underlying layer. Since this lower level service provides transparent data transfer, concerns of the protocol that uses the lower level service can be separated from concerns of the protocol that provides the lower level service. The enhanced functionality accomplished through the interaction of protocol entities in a layer is provided to the next higher layer. Hence, protocol layers only interact directly with adjacent protocol layers in the protocol hierarchy. The principles of layering are described in the OSI-RM ([IS7498:84]). The OSI-RM identifies a fixed number of layers (7), the upper three of which are concerned with application protocols (application-oriented protocols, in OSI terminology). The OSI-RM also outlines the assignment of protocol functions to different layers.
- *application service element composition*: This structuring technique is used in the highest layer of the OSI-RM, the Application Layer, to cope with the broad scope and open-ended nature of this layer. It allows to separate different protocol concerns based on the identification of distinct classes of distributed system applications. Application protocol functions that are to some extent independent of each other are assigned to different Application Service Elements (ASEs). ASEs may be composed in different ways, dependent on the specific class of distributed system applications that is supported. This structuring technique is described in the OSI-ALS ([IS9545:89]).

- *composition of cooperating main service and auxiliary service*: This structuring technique assumes an assignment of application protocol functions to ASEs, as with the previous structuring technique. It is limited, however, to cases where according to the previous technique a hierarchical composition of ASEs would apply. Instead of a hierarchical composition, where one of the ASEs uses the service provided by the other ASE, one of the ASEs directly includes service functions and protocol functions of the other ASE in its own service functions and protocol functions, respectively. The ASE that defines the inclusion (that defines references to service primitives and protocol actions of the other ASE) is called the cooperating main service. The other ASE is called here the auxiliary service (no separate OSI term is available in this case). The application of this structuring technique is rather exceptional. The only application of this technique is currently found in the File Transfer, Access and Management ASE ([IS8571:88]), which describes a possible use of the Commitment, Concurrency and Recovery ASE ([IS9804:90], [IS9805:90]) as auxiliary service.

Each of these structuring techniques is further discussed in a separate subsection below. Related concepts are explained together with the structuring technique in which they are used.

3.1.1 Layer composition

Layering is the basic structuring technique that was used to define the OSI layered protocol architecture (see [Day83] and [Linington83], for example). The overall problem considered by OSI, i.e. the problem of allowing systems to exchange information and to cooperate in distributed applications, is decomposed into manageable pieces, called protocol layers. Each layer is concerned with a layer-specific function which is distributed over all participating systems. The distributed portions of a layer function are assigned to *protocol entities*. A layer *protocol* defines the required interactions of the protocol entities in the layer. The function that results from this interaction and that is offered to a higher layer constitutes a layer *service*. A protocol uses the lower level service, i.e. the service provided by the lower layer, in order to realize the exchange of information between the protocol entities.

In total 7 hierarchically related layers are thus identified in the OSI-RM. The lowest layer is the Physical Layer (layer 1), which uses a combination of available physical media for exchanging information[1]. The highest layer is the Application Layer (layer 7), which does not offer its service to a higher layer[2]. The layered architecture of the OSI-RM is shown in Figure 3.1 (only end-systems, and end-to-end protocols, are shown; a ‘complete’ architecture would also contain intermediate systems).

1. The collection of physical media can also be regarded as a layer (layer 0, to be consistent with the OSI numbering convention), although no protocol can be associated with this layer since its function cannot be distributed over protocol entities.

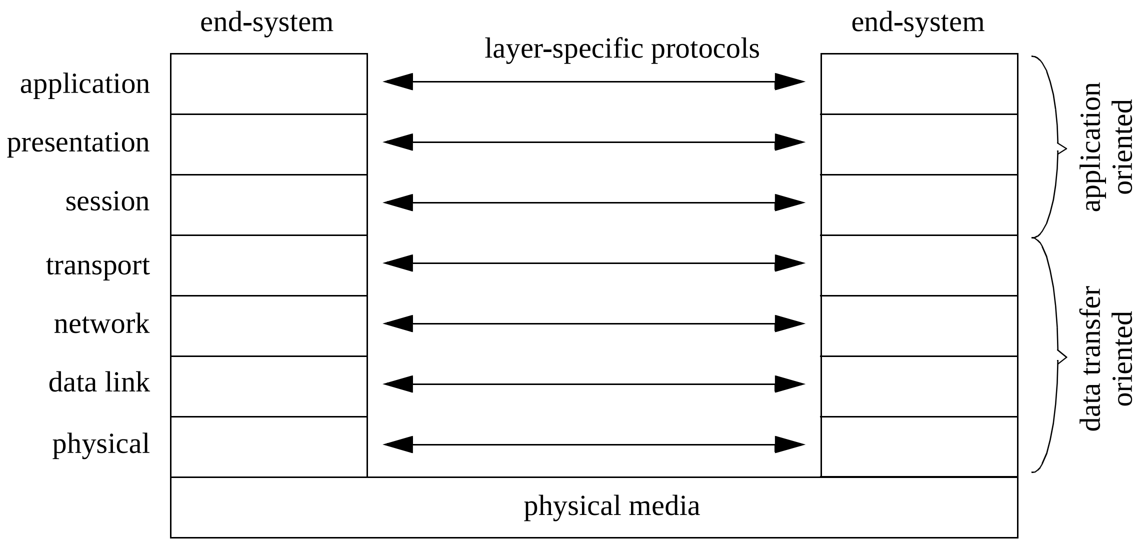

Figure 3.1: Layered architecture of the OSI-RM

A protocol uses the lower level service offered by a lower level protocol in order to implement the service offered to a higher level protocol. Thus the service provided by a protocol is actually the result of the combined behaviour of a stack of protocols. The total functionality of a stack of protocols is assigned to a *service provider.* Since a service provider can be defined at each distinguished level, it is possible to consider an arbitrary service provider, say service provider ‘N’, as the composition of a layer, layer ‘N’, and the underlying service provider, service provider ‘N-1’. This is depicted in Figure 3.2.

The (N)-protocol exchanges units of information, called (N)-*protocol data units* (PDUs), between (N)-protocol entities via the (N-1)-service provider. The use made of the (N-1)-service provider by the (N)-protocol is defined in terms of abstract interactions, called (N-1)-*service primitives* (SPs). A service is thus defined in terms of SPs, and a protocol in terms of PDUs *and* SPs (to be precise, an (N)-protocol is defined in terms of (N)-PDUs, (N)-SPs, and (N-1)-SPs). Whereas PDUs have a defined format and encoding[1], SPs are merely defined as sets of parameters with no prescribed representation. An (N)-PDU is always conveyed in a data parameter of an (N-1)-SP. Since the meaning of

2. The Application Layer can be considered as being ‘open-ended’. This means that new functions will be included when their need has been recognized by the international community. These functions will use existing service functions, both service functions of ASEs in the Application Layer and service functions provided by the Presentation Layer. Also user defined protocols will use these service functions.

1. Except in the Application Layer, where PDUs only have a defined format.

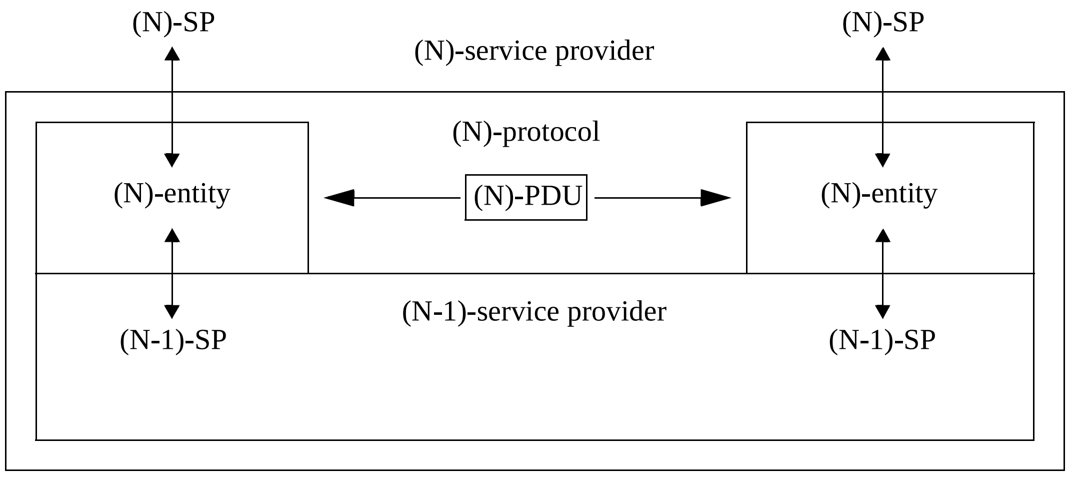

Figure 3.2: Model of an arbitrary layer (layer 'N') in the OSI-RM

an (N)-PDU is only of concern to the (N)-protocol, a data parameter is transferred transparently by the (N-1)-service provider[1], preserving the representation defined by the (N)-protocol. Such a parameter is called an (N)-*service data unit* (SDU). An (N)-PDU normally consists of (N)-*protocol control information* (PCI) and (N)-*user data*. (N)-user data in an (N)-PDU represents the data (corresponding to all or part of an (N)-SDU) that is transparently transferred by the (N)-protocol.

The definition of a protocol in this way permits the implementation of peer protocol entities by different (and isolated) implementation teams. Adjacent protocol entities, however, cannot be implemented in isolation by different implementation teams, unless their abstract interface defined in terms of SPs is first replaced by a concrete interface.

3.1.2 Application service element composition

An *Application Service Element* (ASE) is a separately defined (standardized) part of an application entity. An ASE is always considered together with a peer ASE. The cooperation of these ASEs is defined by an ASE protocol and the function realized by this protocol and offered to a higher level ASE protocol (*not* a higher level layer) is defined by an ASE service. The definition of an ASE protocol and an ASE service is done in the same way as the definition of a 'normal' protocol and service. An important distinction, however, is the scope of an ASE protocol and an ASE service. An ASE protocol is concerned with a particular part or aspect of the interaction of application entities, and is

1. An exception is the Presentation Service provider. The Presentation Layer is responsible for the coding of application PDUs in transit between end-systems. The Presentation Service provider preserves the meaning of information during transfer, not necessarily the representation of the information.

therefore not necessarily involved in all information exchanges between these application entities.

Consequently, an ASE protocol may sometimes use several lower level services, and an ASE service may be defined as being part of a more comprehensive application service. Figure 3.3 depicts an example of a composition of different ASE protocols.

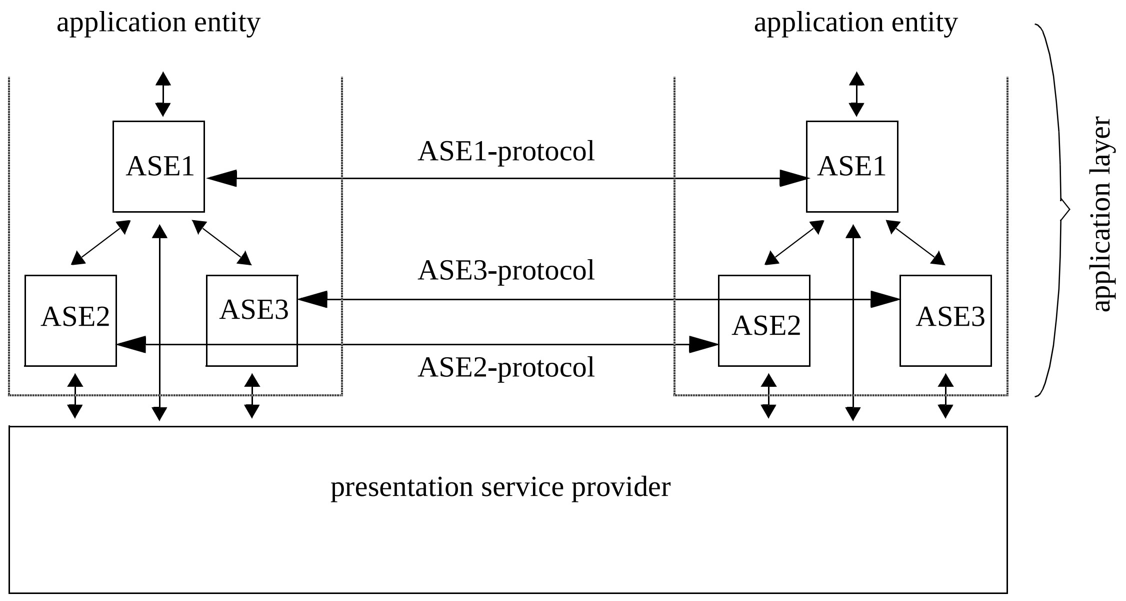

Figure 3.3: Example of application service element composition

The potentially complex relationships between ASE protocols cannot be represented by the technique of layer composition alone. Consider, for example, the composition depicted in Figure 3.3. If ASE1 is involved in request service primitives with successively ASE2, the underlying presentation entity, and ASE3, it is assumed that its peer ASE participates in corresponding indication service primitives in the same order. However, if no control is exercised on the ordering of service primitives in which ASE2, ASE3 and the presentation entities are involved it is unlikely that this order will be preserved.

An auxiliary element is introduced for this purpose, called the *Single Association Control Function* (SACF). A SACF represents the rules for controlling the relationship between service primitives in which ASEs are involved and service primitives in which the presentation entities are involved. The relationship between these service primitives must be such that the order in which ASEs send (receive) information through service primitives corresponds to the order in which the presentation entities are involved in service primitives that convey this information. Notice that there must be a specific SACF

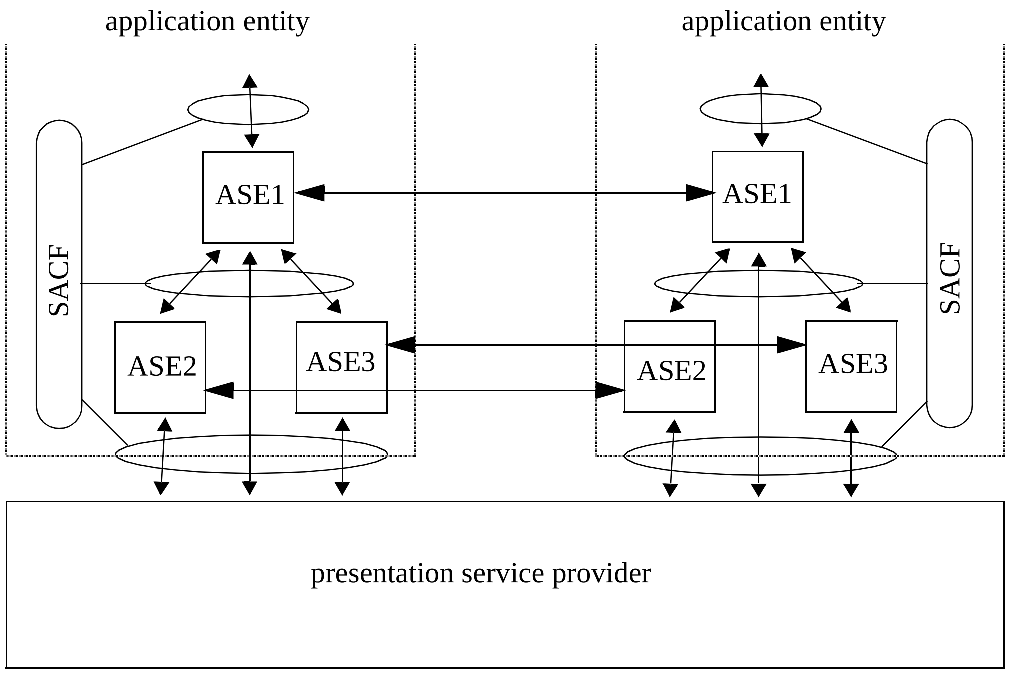

Figure 3.4: SACF for a specific combination of ASEs

for each combination of ASEs. Figure 3.4 depicts an example of a SACF and a combination of ASEs.

ASE protocols only define the cooperation of two peer ASEs, i.e. they are limited to peer-to-peer relationships. Many distributed applications, however, require multi-peer relationships between application entities. For this reason another element is introduced, called the *Multiple Association Control Function*[1] (MACF). A MACF locally relates and constrains multiple point-to-point application services, each of which is provided by one or more ASE protocols that all use the same presentation connection. In addition, a MACF may interact with peer MACFs to perform distributed coordination of application services[2]. One or more MACFs provide an application service that can be used for multi-peer distributed processing. Note that there must be a specific MACF for each combination of lower level application services. Figure 3.5 depicts an example of a MACF that locally

1. SACF and MACF are concepts defined in the first version of the Application Layer Structure ([IS9545:89]). A revision of this standard defines the more general concept of Control Function. We use here the original ALS concepts since these have been used in the Application Layer standards considered in this chapter.

2. Whether the interaction between peer MACFs should be defined by a MACF protocol is not clear from the Application Layer Structure. The only MACF that is explicitly defined so far (the MACF for Distributed Transaction Processing, in [IS10026:92]), is only concerned with local coordination.

coordinates the use of two lower level application services and coordinates its own operation with two peer MACFs (assuming the definition of a MACF protocol).

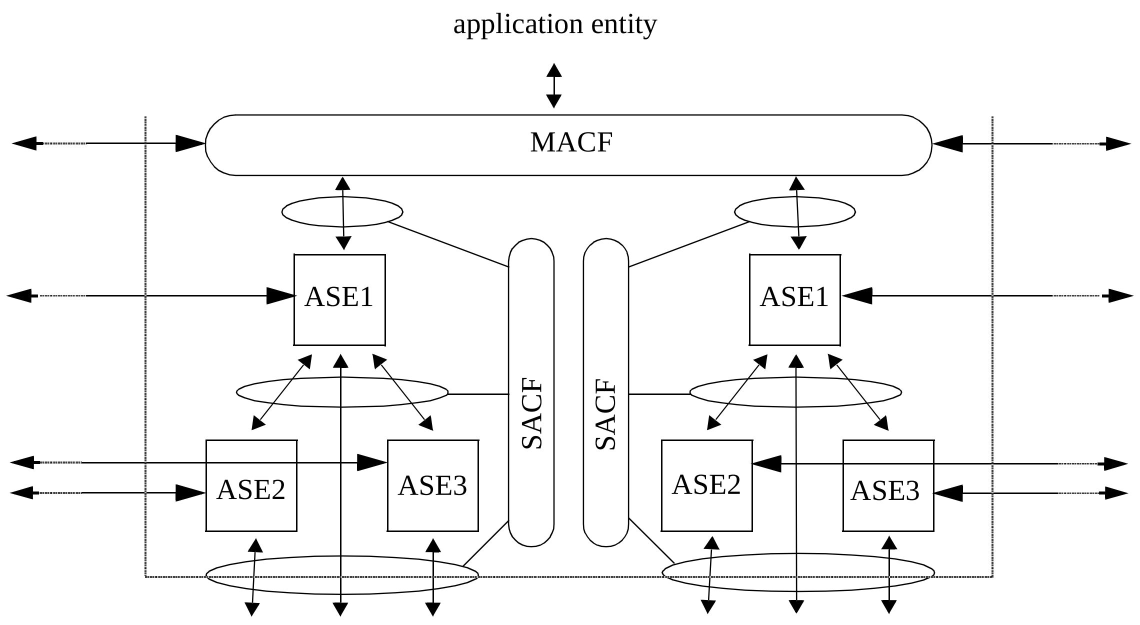

Figure 3.5: Example of control functions required by application service element composition

A specific SACF and a specific MACF may be defined in an application protocol framework standard. A framework standard defines a specific ASE and defines how other ASEs, defined in other standards, may be used in combination with this ASE. An example of an application protocol framework standard is the Distributed Transaction Processing standard ([IS10026:92]).

3.1.3 Composition of cooperating main service and auxiliary service

The *cooperative main service* approach is not defined in general terms by the OSI-RM or the OSI-ALS. The following is an attempted generalization of the cooperative main service approach, based on the only present application of this approach, namely the inclusion of the Commitment, Concurrency and Recovery Service Element ([IS9804:90], [IS9805:90]) by the File Transfer, Access and Management standard ([IS8571:88]). For convenience we introduce the following terms:

- *main ASE*: the cooperating main service (element); and
- *auxiliary ASE*: the ASE that is referenced by the main ASE.

We will use the prefixes ‘main’ and ‘auxiliary’ consistently in order to indicate aspects of the main ASE and the auxiliary ASE, respectively.

The main ASE service defines which main service primitives are able to carry the semantics of an auxiliary service primitive and which parameter of these main service primitives is used for this purpose. The mapping between main service primitives and auxiliary service primitives must be such that the constraints that are defined for the main ASE service and the auxiliary ASE service do not conflict. For example, the ordering of auxiliary service primitives may not conflict with the ordering of main service primitives that carry the semantics of these auxiliary service primitives. The interpretation of the main ASE service is that of a conjunction of the constraints defined on main service primitives (in the main ASE service) and the constraints defined on the auxiliary service primitives (in the auxiliary ASE service).

Similarly, the main ASE protocol defines which main PDUs are able to carry the semantics of an auxiliary PDU and which field in these main PDUs is used for this purpose. The format of a main PDU that carries an auxiliary PDU is given by the format defined in the main ASE protocol, with the field that carries the auxiliary PDU replaced by the format of this PDU (as defined in the auxiliary ASE protocol). The constraints and actions defined in the main ASE protocol may not conflict with those defined in the auxiliary ASE protocol. The interpretation of the main ASE protocol is that of the conjunction of constraints and the cumulation of actions defined in the main ASE protocol and the auxiliary ASE protocol.

Figure 3.6 depicts an example of the composition of a main ASE (the cooperating main service) and an auxiliary ASE.

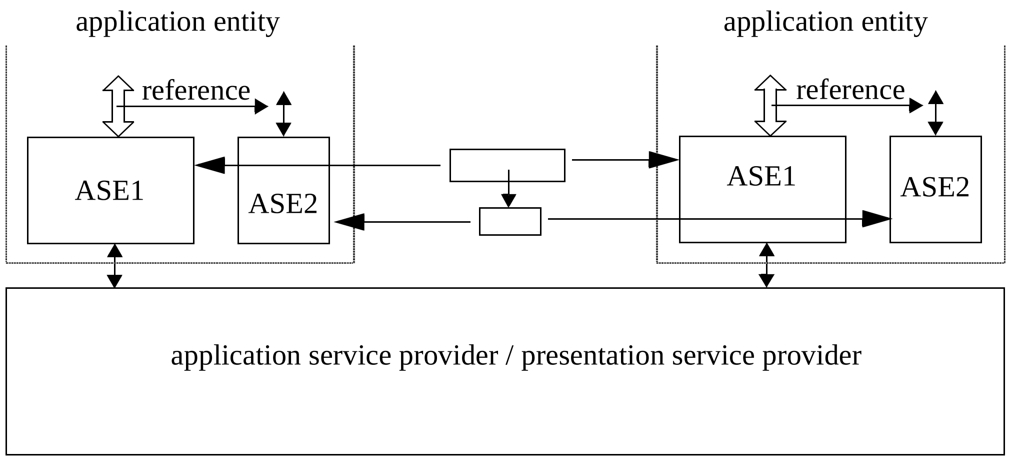

Figure 3.6: Example of the composition of a main ASE (ASE1) and an auxiliary ASE (ASE2)

3.2 Application protocol standards

Although the development of application protocols was initially lagging behind the development of data transfer protocols, at present a rich set of application protocol standards is available that provides support to a wide variety of distributed system applications. Figure 3.7 presents an overview of the most important application protocol standards (each with an associated service standard) that are currently completed. It also shows the relationships between these standards.

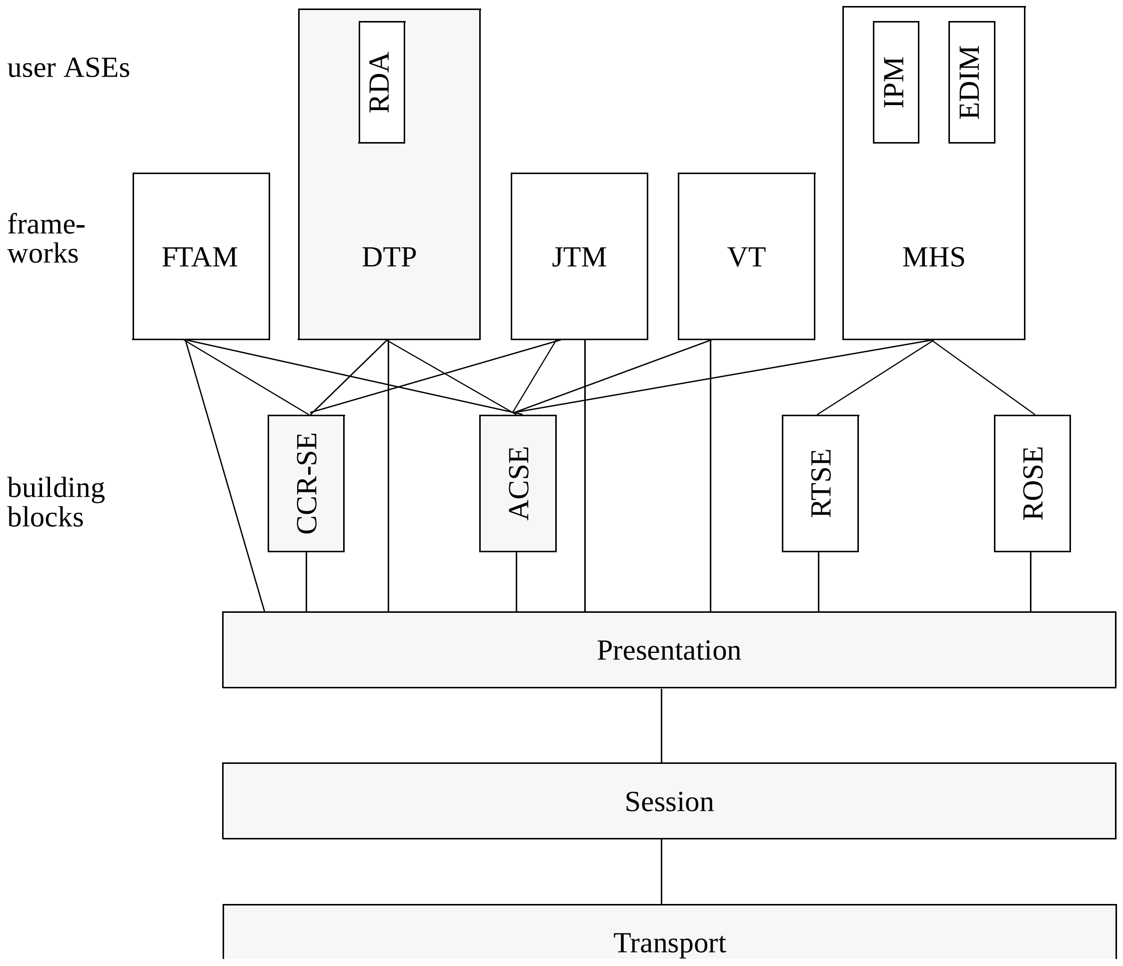

Figure 3.7: Application protocol standards

All application protocols use, directly or indirectly, the service provided by the Transport Layer. The connection oriented Transport Service ([IS8072:86]) is a relatively simple service that comprises the following functions:

- *connection establishment*: this function provides confirmed connection establishment that allows the negotiation of quality characteristics (‘quality of service’) of the connection, including the expedited data transfer option (see below).
- *data transfer*: there are two data transfer functions, viz. ‘normal’ and ‘expedited’ data transfer. Normal data transfer is always available and provides in-sequence transfer of transport SDUs (TSDUs). Expedited data transfer is only available if selected during connection establishment. It provides in sequence transfer of expedited TSDUs (of limited length). Expedited transfer is never slower than normal transfer, and possibly, but not necessarily, faster than normal transfer.
- *connection release*: this function provides unconditional release of the connection.

In addition, a connectionless-mode Transport Service is defined (in an addendum to [IS8072:86]) which comprises a single function: the transfer of self-contained TSDUs. With this type of service, TSDUs are not related and the transfer of TSDUs does not require previously established arrangements of the service provider and the service users.

All application protocols indicated in Figure 3.7 offer a connection oriented (or association oriented) service, except the Message Transfer (MT) service of the Message Handling System (MHS) standard, which provides a (kind of) connectionless-mode service. In addition, connectionless-mode services, and supporting protocols, are defined for the Session Layer, the Presentation Layer, and the Association Control Service Element (ACSE). These services are very similar to the connectionless-mode Transport Service. Until now, none of the application protocol framework standards makes use of lower level connectionless-mode services.

It is not possible to give a comprehensive overview of all present application service and protocol standards. We will therefore consider a representative ‘profile’ of the OSI-ULA, which permits an illustration of the subdivision of the application protocol functionality and which can be used as a reference for evaluation and comparison in later chapters. This profile is indicated in Figure 3.7 by the shaded boxes, and consists of:

- the Session Layer standards ([IS8326:87], [IS8327:87]) and Presentation Layer standards ([IS8822:88], [IS8823:88]): these standards illustrate the layering approach. The Presentation Protocol uses the Session Service, and Application Layer protocols use the Presentation Service.
- the Association Control Service Element (ACSE, [IS8649:88], [IS8650:88]) and the Commitment, Concurrency and Recovery Service Element (CCR-SE, [9804:90], [IS9805:90]): these standards are examples of the most general building blocks of the Application Layer. The ACSE protocol and the CCR-SE protocol only use the Presentation Service, while the ACSE service and the CCR-SE service are used by many

other Application Layer protocols and thus support many distinguished classes of distributed system applications.

- the Distributed Transaction Processing (DTP) standard ([IS10026:92]): this standard is an example of an application protocol framework standard. It defines an ASE for the support of distributed transaction processing, the TP-ASE, and defines how this ASE can be used in combination with lower level ASEs and with ‘user’ ASEs. A user ASE is an ASE which can fill certain functional holes in the framework standard. There may be several ASEs which can perform the role of user ASE in a framework standard. The DTP standard includes the definition of a SACF (since it must define the combination of the TP-ASE and lower level ASEs) and of a MACF (since it is concerned with multi-peer interactions).

These standards are further explained in the following sections.

3.3 Session Layer

The main functions of the Session Layer are support of the synchronization needs of application entities and support of the controlled use of transfer facilities by application entities. *Synchronization* entails the following functions:

- transfer of synchronization points (identified by serial numbers) for synchronization and resynchronization purposes;
- generating serial numbers of synchronization points (except when the application entities want to agree upon a new initial synchronization point);
- separation of data sent before and after a synchronization point (in case of synchronization) and purging of undelivered data sent before a synchronization point (in case of resynchronization); and
- signalling that an identified activity of an application entity is started, interrupted, resumed, discarded, or ended.

Figure 3.8 illustrates the separation and purging of data accomplished by the Session protocol.

The *controlled use of transfer facilities* (dialogue control) entails the following functions:

- management and transfer of tokens (representing the exclusive right to invoke a specific service); and
- constraining the use of services according to the token states.

Figure 3.9 illustrates these functions.

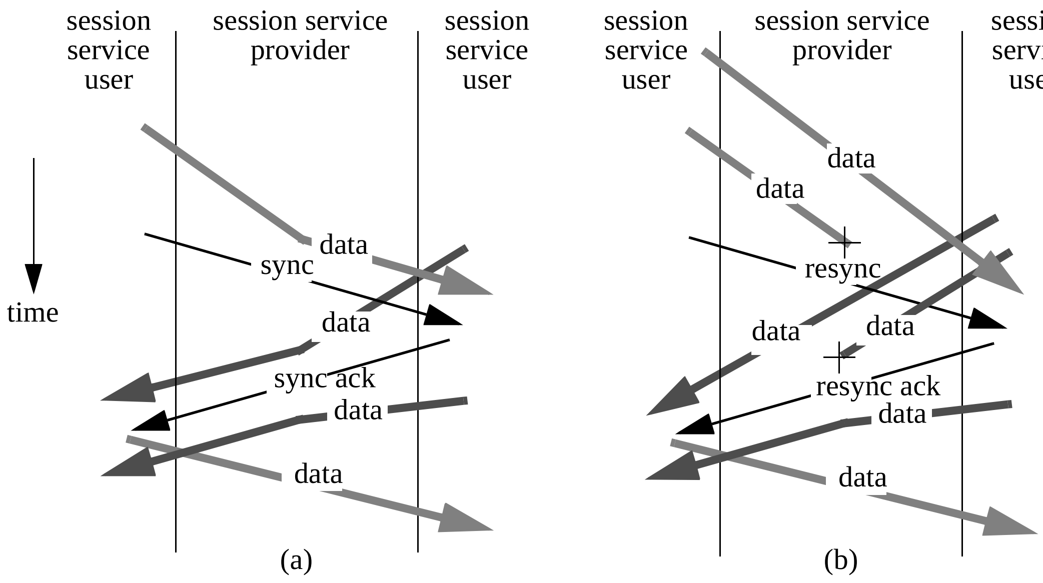

Figure 3.8: Separation of data (a) and purging of data (b)

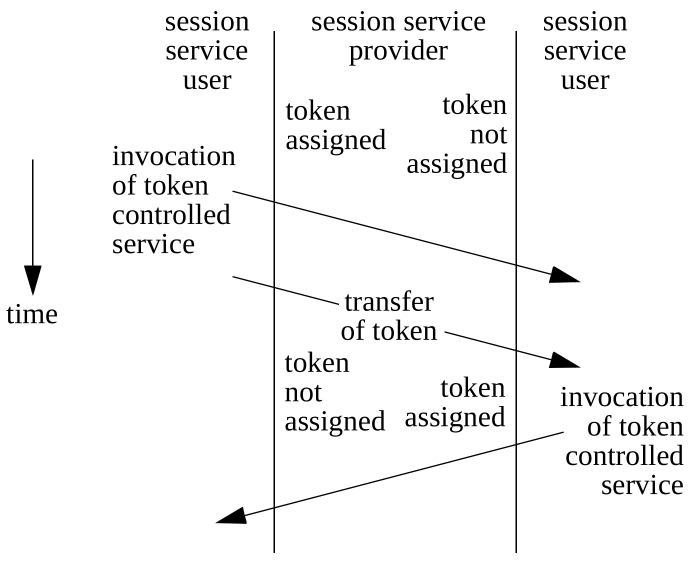

Figure 3.9: Use of a token controlled service

Other functions of the Session Layer include:

- establishment of a suitable session connection:

  - negotiation of session functionality, quality of service, and, if required, initial synchronization points and initial token settings;
  - establishing or re-using a suitable transport connection; if a new transport connection is established to support a session connection, the transport connection must have been established before the above mentioned negotiations can start (i.e., the session and transport connection are not established in ‘one shot’).
- matching and exploiting data transfer capabilities of the transport service:
  - segmentation/reassembly of session SDUs;
  - concatenation/separation of session PDUs.
- termination of the session connection without loss of data (orderly release), or with possible loss of data (abort):
  - termination or retaining the transport connection; if the transport connection is terminated, this must occur before the termination of the session connection can start.

Establishing, retaining, re-using, and terminating transport connections are internal functions, i.e. functions of the Session Protocol that have no visibility in the Session Service. Also segmentation/reassembly and concatenation/separation are internal functions. All other functions are visible in the Session Service.

The Session Service provides a *kernel* functionality consisting of the connection establishment function, the normal data transfer function, and the connection termination functions (including orderly release). All other functions of the Session Service are *optional* and have to be negotiated between the Session Service users. The Session Protocol only monitors this negotiation, i.e. it must support whatever functionality is agreed upon by the Session Service users. The session connection establishment function also comprises the negotiation of quality of service (QoS). Only two QoS parameters are of relevance to the Session Protocol:

- optimized dialogue control: a boolean which indicates whether (extended) concatenation of session PDUs should be performed; and
- extended control: a boolean which indicates whether the transport expedited flow option should be used for bypassing normal flow control.

All other QoS parameters are negotiated between the application entities and the Transport Service provider, and are thus merely passed through by the Session Protocol.

3.4 Presentation Layer

The Presentation Layer is concerned with the *representation of information in transit* between two application entities. It is not concerned with the interpretation of the information. The need for the presentation layer functionality stems from the fact that in an ‘open’ and thus potentially heterogeneous environment different conventions for the local representation of information may be used by different computer systems. A common representation must be agreed if information is exchanged between two systems. If the local representation used by a system differs from the agreed common representation, a translation to (at the sending side) or from (at the receiving side) the common representation must be performed.

An application PDU (APDU) is mapped onto a Presentation Data Value (PDV) in a presentation SDU (PSDU). Potentially multiple PDVs may be contained in an PSDU (thus permitting a kind of concatenation/separation at service level). Since the Application Layer is relieved from concerns about the common representation of information, and local representations are not within the scope of protocol standardization, Application Layer protocols define APDUs independent of any specific representation (using an abstract syntax notation). A set of APDUs associated with a specific Application Layer protocol constitutes the *abstract syntax* of that protocol. A specific representation of information units is called a concrete syntax. Thus, implementations of application entities will use a local concrete syntax to represent the APDUs, and a common representation used by the Presentation Protocol constitutes a concrete transfer syntax, or *transfer syntax* for short. Figure 3.10 depicts the relationship between an abstract syntax, local concrete syntax, and transfer syntax.

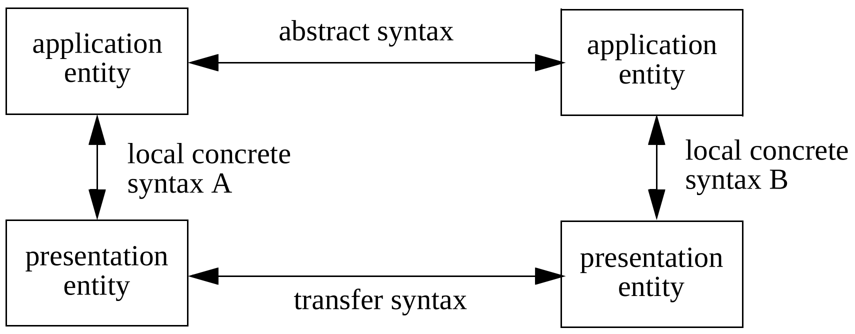

Figure 3.10: Relation between an abstract syntax, local concrete syntax, and transfer syntax

The main function of the Presentation Service is the transfer of information, as PDVs in PSDUs, independent of the local concrete syntaxes used for the representation of PDVs.

The main functions of the Presentation Protocol are:

- negotiation of transfer syntaxes to support the abstract syntaxes used by the application entities; and
- transformation from the local concrete syntax to the transfer syntax, and vice versa.

For each abstract syntax that is used by the application entities, a suitable transfer syntax must be determined by the Presentation Protocol. The negotiation of abstract syntaxes (between application entities) is performed in conjunction with the negotiation of transfer syntaxes (between presentation entities)[1]. This is possible during presentation connection establishment. Alternatively, the use of an optional service function, the *context management* function, can be selected by the application entities, which permits the addition or deletion of abstract syntaxes and supporting transfer syntaxes after the presentation connection has been established.

The coupling of an abstract syntax and a transfer syntax is called a *presentation context*. At each end of the presentation connection, the application entity and the presentation entity refer to a presentation context by means of a presentation context identification (PCId). The PCId of a presentation context is locally agreed between the application entity and the presentation entity during the negotiation of that context[2]. A PCId is necessary each time a PDV is passed from the application entity to the presentation entity (to permit the presentation entity to determine the associated transfer syntax), and vice versa (to permit the application entity to determine the associated abstract syntax).

The Presentation Service also provides all the functions of the Session Service (which are thus passed through by the Presentation Protocol), with the only difference that the Presentation Protocol determines the representation of PDVs contained in the user data parameter of Session Service primitives. Besides context management, application entities may also select a *context restoration* function (only if context management is also selected). When context restoration is selected, presentation entities record the presentation contexts in force each time a synchronization point is transferred between the application entities. The presentation contexts associated with a particular synchronization point are restored by the presentation entities if the application entities request resynchronization or activity resumption from this point.

1. Although the presentation protocol is not involved in the negotiation of abstract syntaxes, it must be informed of the set of abstract syntaxes that is being negotiated in order to perform the transfer syntax negotiation.

2. Presentation context identifications used at different ends of the presentation connection are related by presentation context identifiers exchanged in the Presentation Protocol.

The establishment and termination of the presentation connection is performed in conjunction with the establishment and termination of the session connection (‘one shot’ establishment and termination).

3.5 Application Layer building blocks

A number of Application Layer protocols are defined that act as general building blocks for other Application Layer protocols. At present, four such building blocks are defined. We only consider the building blocks that are used by the Distributed Transaction Processing standard, described in section 3.6.

3.5.1 Association Control

The Association Control Service Element (ACSE) defines functions for the establishment and termination of an association between two application entities, called an application association. An application association is a mutual agreement of two application entities concerning their cooperative relationship, which comprises, among others, the identification of ASEs that must be available to support the cooperation.

The ACSE provides the possibility for an application entity to respond to a request for cooperation from any other application entity. If the necessary ASEs are available in the addressed application entity, the response from the application entity can in principle be affirmative, leading to the establishment of an appropriate application association. If this is not the case, the response will be negative, and no application association will be established. This functionality is essential in an open environment, and must therefore always be present in any application entity.

An application association is established and terminated in conjunction with a presentation connection and a session connection. The possibility of negotiating or selecting optional functions of the Presentation Service and the Session Service is offered via the ACSE service to the initiator and acceptor of the application association.

3.5.2 Commitment, Concurrency and Recovery

The Commitment, Concurrency and Recovery Service Element (CCR-SE) defines functions for the coordination of two application entities such that the activity in which these application entities participate has atomic action properties. An activity with atomic action properties is called an *atomic activity.* Atomic action properties are:

- atomicity: the operations that are part of the atomic activity are either all performed or none of them is performed;
- consistency: the operations are correctly performed[1];

- isolation: intermediate results of the atomic activity are not accessible, except by operations of the activity itself;
- durability: the final results of the atomic activity are not effected by system or communication failures.

The operations of an atomic activity may access and manipulate different kinds of resources (e.g. information resources and physical objects).The collective state of the resources which is used by the atomic activity is referred to as *bound data*. The coordination provided by the CCR-SE ensures that bound data can either be released in the initial state (the state that existed before the start of the activity) or in the final state (the state that results from a successful performance of all operations) at the end of the atomic activity, and that the intermediate states of the bound data are not visible outside the atomic activity. This is illustrated in Figure 3.11. The operations that effect bound data and the (possible) changes effected are not considered by the CCR-SE.

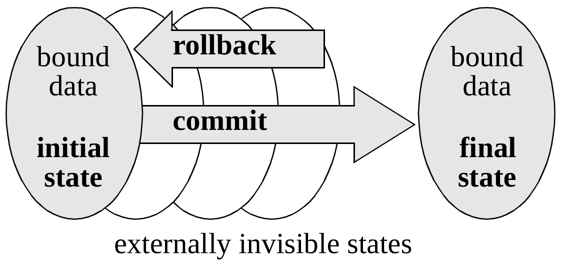

Figure 3.11: Transition of bound data in an atomic activity

The state and control data about an atomic activity that is maintained for recovery purposes is called *atomic action data*. The CCR-SE defines when atomic action data must be recorded, or logged, such that this data can survive system and communications failures.

The relationship between two CCR-SE service users is characterized as a superior-subordinate relationship. The superior user normally takes initiative for the coordination supported by CCR-SE. The coordination provided by the CCR-SE is based on a two-phase commit mechanism with presumed rollback. Two-phase commitment consists of two phases:

- voting phase: the subordinate is asked by the superior whether it can release its bound data in the final state (commitment offer);

1. Both according to the their specification and not conflicting consistency conditions which may be defined for the application.

- commitment or rollback phase: the superior orders the subordinate to release its bound data in the final state (commitment)[1] or to release its bound data in the initial state (rollback).

Presumed rollback implies an assignment of recovery responsibility to the CCR-SE service users such that the overhead for logging and removing logged data can be minimized in an implementation.

The CCR-SE assumes the availability of an application association which supports at least the exchange of CCR PDUs. This association must have been established, using the ACSE service, by another ASE. The use of the CCR-SE therefore always requires one or more other ASEs (besides the ACSE), defined by a framework standard that incorporates the CCR-SE as a building block ASE.

3.6 Distributed Transaction Processing

The Distributed Transaction Processing (DTP) standard defines functions that support the performance of transactions by two *or more* application entities. A transaction is an atomic activity, with the atomic properties mentioned in section 3.5.

DTP distinguishes three types of peer-to-peer relationships between application entities:

- *application association*: Application associations are established and terminated using the ACSE service. An application association may be idle, i.e. not in use by application entities performing DTP functions. DTP does not define when and how application associations are established.
- *dialogue*: A dialogue is a relationship between two application entities that perform DTP functions. A dialogue may survive system and communication failures (although it is currently not defined what atomic action data must be maintained). The exchange of information between application entities may be temporarily blocked because of such failures. The exchange of information on a dialogue is supported by an application association. Several application associations may be used consecutively by a dialogue (in case of communication failures).
- *transaction branch*: A transaction branch is a relationship between two application entities that perform DTP functions concerned with transaction coordination. A transaction branch implies the existence of a dialogue. Multiple consecutive transactions can be supported on a dialogue.

1. A commitment offer from the subordinate does not imply that the superior will order commitment. The atomic activity considered in the scope of CCR may be part of a larger atomic activity, involving many other application entities, each of which may be required to be able to release bound data in the final state before an order of commitment is given.

Multi-peer relationships between application entities can be built with dialogues and with transaction branches. In both cases these are modelled as trees, with application entities as nodes and dialogues or transaction branches as arcs. A *transaction tree* is by definition part of a *dialogue tree* (since a transaction branch requires a dialogue). DTP allows the use of different parts of a dialogue tree for independent transactions. Figure 3.12 illustrates the relationship between a dialogue tree and multiple transaction trees.

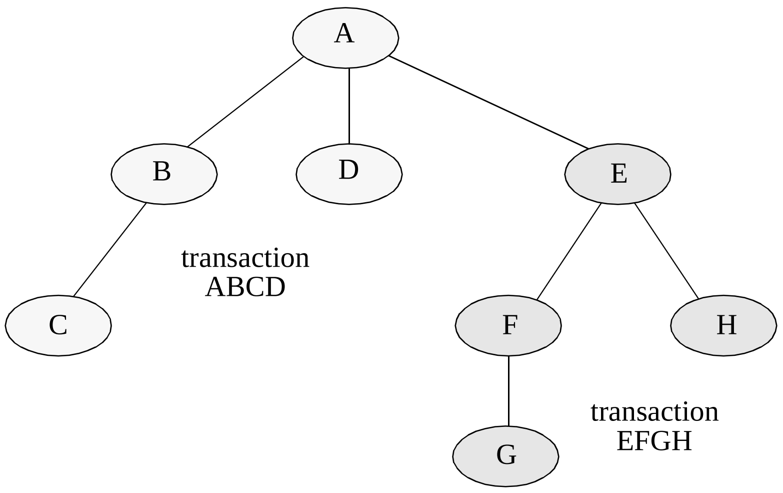

Figure 3.12: Dialogue tree supporting two concurrent transactions

The functions of DTP can be broadly classified as either being concerned with dialogues (on which transactions can be performed) or with transactions (on dialogues). Dialogue related functions include:

- establishment and termination of a dialogue between application entities (thus changing the dialogue tree);
- supporting the orderly use of the dialogue by application entities (requesting and granting control of the association);
- supporting the synchronization of processing activities of application entities (hand-shaking).

Transaction related functions include:

- *begin* transaction function: initiating a transaction branch between two application entities (thus changing the transaction tree).

  If the initiating application entity is already involved in a transaction, the transaction tree is extended, with the initiating application entity acting as superior for this new transaction branch and the accepting application entity acting as subordinate for this branch. Otherwise, if the initiating entity was not involved in a transaction, a transaction tree is created consisting solely of this new transaction branch, with the initiating entity acting as the master of the transaction.

This function is only available if the ‘unchained transactions’ option has been selected. This option implies that transaction trees must be explicitly built in order to perform a transaction and will dissolve again after termination of the transaction. When the unchained transactions option is selected it is possible that at a given point in time two users of the dialogue are involved in the same transaction, in different transactions, or one or both of them are not involved in any transaction. If the unchained transactions option is not selected, the ‘chained transactions’ option must be selected. This option implies that all dialogue users are always involved in the same transactions. When a transaction terminates, a next transaction is implicitly started. Only when the dialogue is terminated, a transaction branch is pruned from the transaction tree. The following functions are available with either of the options.

- coordination functions:
  - *prepare* function: allows a superior to request completion of transaction processing in a transaction sub-tree (coordinated by a subordinate of this superior). All application entities in this sub-tree are requested to prepare the release of their bound data in the final state.
  - *ready* function: allows a subordinate to indicate to its superior that all application entities in the transaction sub-tree are ready to release their bound data in the final state.
  - *commit* function: allows a user to indicate that it has completed all processing for the current transaction and to request termination of the transaction with the release of bound data in the final state. When the user is a master, this function initiates the termination of the transaction; otherwise, this function is used to propagate termination of the transaction.
  - *rollback* function: allows a user to request the termination of the transaction with the release of bound data in the initial state.
  - *done* function: allows a user to indicate that is has released its bound data.
  - commitment/rollback *complete* function: this function is initiated by the DTP service provider to indicate to all users involved in the transaction that commitment/ rollback is complete.

A dialogue only survives failures if it supports a transaction. Otherwise, a failure will cause an abort of the dialogue. Furthermore, a dialogue can only be terminated if no transaction is in progress. A transaction branch can only be initiated on a dialogue if no current transaction branch exists on the dialogue. Termination of a dialogue may result in a new, pruned dialogue tree (if the dialogue was connected to a ‘leaf’ in the original dialogue tree), or two new dialogue trees (if the dialogue connected two ‘non-leafs’ in the original dialogue tree). If the chained transactions option has been selected on the dialogue, similar changes apply to the transaction tree.

DTP is a rather complex standard. It includes the definition of a TP-ASE, a TP-SACF, and a TP-MACF. In addition, it uses the ACSE service, the CCR-SE service (optionally), and the Presentation Service (and via the Presentation Service, also the Session Service). Finally, it must be associated with one or more (undefined) user ASEs. The DTP standard therefore defines a number of internal services in order to facilitate the definition of the protocol. Figure 3.13 depicts the TP protocol entity architecture. This architecture also

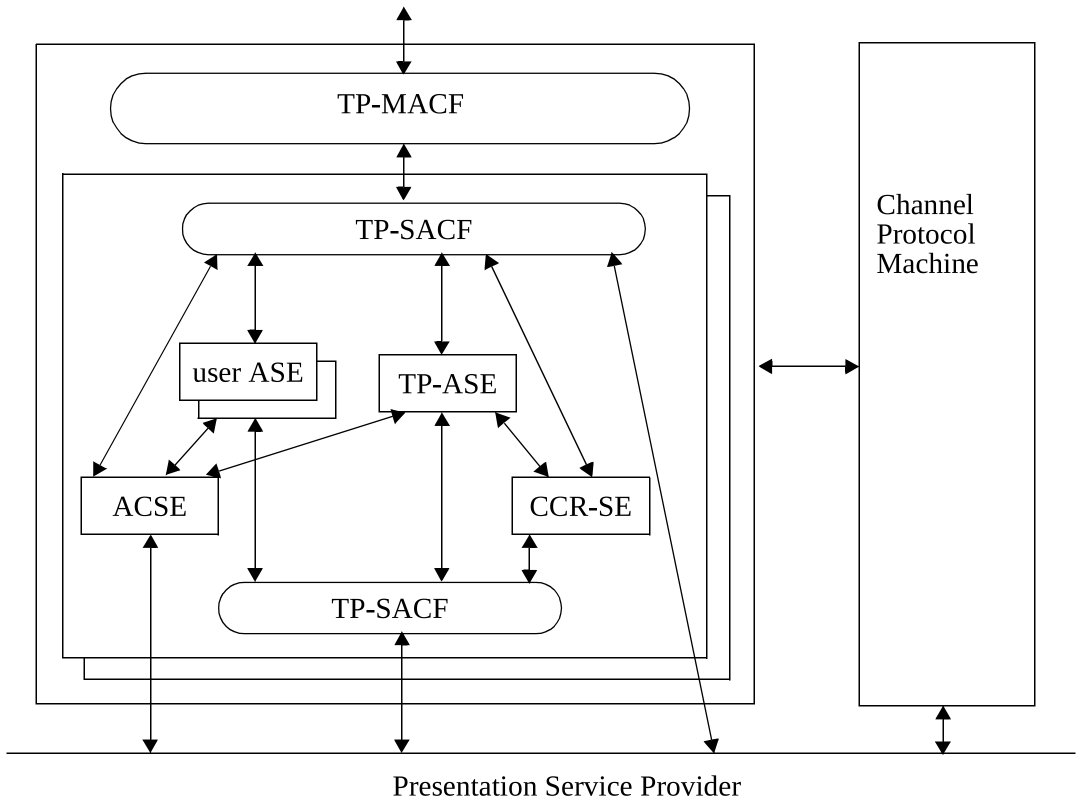

Figure 3.13: Architecture of a Transaction Processing protocol entity

includes a Channel Protocol Machine (CPM) which establishes and terminates ‘channels’ as appropriate for the purpose of recovery. The CPM includes the same elements as the Transaction Processing Protocol Machine (TPPM), except that no user ASEs are included. The TPPM is the only ‘user’ of the CPM.

3.7 Conclusion

The OSI Upper Layer Model (OSI-ULM) comprises three structuring techniques that are used to compose application protocol standards:

- layer composition;
- Application Service Element (ASE) composition; and
- composition of a cooperating main service and one or more auxiliary services.

Layer composition yields a vertical structure. Each layer protocol uses the service provided by the next lower layer and provides an enhanced service to the next higher layer. The OSI Upper Layer Architecture (OSI-ULA) defines three layers: the Session Layer, the Presentation Layer, and the Application Layer. The Session Layer and the Presentation Layer are each defined by one service and one protocol standard. The application layer is, however, further structured, according to the second (as a rule) and third (as an exception) mentioned structuring technique.

The Session Service and Protocol and also the Presentation Service and Protocol are structured in terms of functional units. A service functional unit is a grouping of related service functions (service elements) and a protocol functional unit is a grouping of protocol functions (elements of procedure). Usually, only one functional unit in the service and protocol is always selected. This functional unit is called the kernel functional unit. All other functional units have to be negotiated between the service users. The main purpose of functional units is to allow application entities to select the functions they actually require.

ASEs can be considered as functional units of ‘sub-layers’ of the Application Layer. The only, but important, difference is that ASEs are defined in separate (service and protocol) standards. Hence, the relationship between ASEs that are used ‘next to each other’ in the same sub-layer is not defined by these ASEs. An auxiliary element, called the Single Association Control Function (SACF), is introduced to define necessary relationships that characterize the desired composition. Since ASEs, like the Session Layer and the Presentation Layer, only cover peer-to-peer interaction, and because many distributed system applications require multi-peer interactions, yet another element is introduced. This element is called the Multiple Association Control Function (MACF). A MACF defines the local coordination of multiple application associations and possibly also the interactions between more than two application entities. A MACF uses the services of one or more ASEs and possibly also the Presentation Service. SACFs and MACFs are usually defined in so called application protocol framework standards, which define an application-specific ASE and which include references to one or more (building block) ASEs whose services can optionally be used by the application-specific ASE. The Distributed Transaction Processing standard is an example of a framework standard.

A cooperating main service can be considered as a ‘main’ ASE with ‘holes’ in its service and protocol that must be filled by an ‘auxiliary’ ASE. Thus, the main ASE service defines the relationship between its service functions with ‘dummy’ parameters and service functions of the auxiliary ASE service that are used to replace the dummy parameters. Similarly, the main ASE protocol defines the relationship between its PDUs with ‘dummy’ fields and PDUs of the auxiliary ASE protocol that are used to replace the dummy fields. The advantage of this composition is that it does not require the introduction of a SACF, since the relationship between the main ASE and the auxiliary ASE is defined by the main ASE. The disadvantage is that the composition possibilities are rather restricted.

The layering approach and the composition of ASEs is illustrated with descriptions of the Session Layer, the Presentation Layer, the Association Control Service Element, the Commitment, Concurrency and Recovery Service Element, and the Transaction Procession Application Service Element.

References

[Day83] Day, J.D., and Zimmermann, H., The OSI reference model, *Proceedings of the IEEE,* Vol. 71, No. 12, December 1983, 1334-1340.

[IS7498:84] ISO, *Open systems interconnection - basic reference model,* International Standard ISO 7498 Part 1, 1984.

[IS8072:86] ISO, *Transport service definition*, International Standard ISO 8072, 1986.

[IS8326:87] ISO, *Basic connection oriented session service definition,* International Standard ISO 8326, 1987.

[IS8327:87] ISO, *Basic connection oriented session protocol specification,* International Standard ISO 8327, 1987.

[IS8571:88] ISO, *File transfer, access and management,* Part 1: General introduction, Part 2: Virtual filestore definition, Part 3: File service definition, Part 4: File protocol specification, International Standard ISO 8571, 1988.

[IS8649:88] ISO, *Service definition for the association control service element,* International Standard ISO 8649, 1988.

[IS8650:88] ISO, *Protocol specification for the association control service element,* International Standard ISO 8650, 1988.

[IS8822:88] ISO, *Connection oriented presentation service definition,* International Standard ISO 8822, 1988.

[IS8823:88] ISO, *Connection oriented presentation protocol specification*, International Standard ISO 8823, 1988.

[IS9545:89] ISO, *Application layer structure*, International Standard ISO 9545, 1989.

[IS9804:90] ISO, *Service definition for the commitment, concurrency and recovery service element*, International Standard ISO 9804, 1990.

[IS9805:90] ISO, *Protocol specification for the commitment, concurrency and recovery service element*, International Standard ISO 9805, 1990.

[IS10026:92] ISO, *Distributed transaction processing*, Part 1: OSI TP model, Part 2: OSI TP service, Part 3: OSI TP protocol, International Standard ISO 10026, 1992.

[Linington83] Linington, P.F., Fundamentals of the layer service definitions and protocol specifications, *Proceedings of the IEEE*, Vol. 71, No. 12, December 1983, 1341-1345.