The OSI Reference Model

JOHN D. DAY AND HUBERT ZIMMERMANN

Invited Paper

Abstract—**The early successes of computer networks in the mid-1970's made it apparent that to utilize the full potential of computer networks, international standards would be required. In 1977, the International Standards Organization (ISO) initiated work on Open Systems Interconnection (OSI) to address these requirements. This paper briefly describes the OSI Reference Model. The OSI Reference Model is the highest level of abstraction in the OSI scheme. The paper first describes the basic building blocks used to construct the network model. Then the particular seven-layer model used by OSI is briefly described, followed by a discussion of outstanding issues and future extensions for the model.**

I. Introduction

IN THE mid-1970's, the development and use of computer networks began to achieve considerable attention. The early successes of the ARPANET [1] and CYCLADES [2], the immediate commercial potential of packet switching, satellite and local network technology, and the declining cost of hardware made it apparent that computer networking was quickly becoming an important field. It was also apparent that to utilize the full potential of computer networks, international standards to ensure interworking would be required.

In 1978, the International Organization for Standardization (ISO) Technical Committee 97 on Information Processing, recognizing that standards for networks of heterogeneous systems were urgently required, created a new subcommittee (SC16) for "Open Systems Interconnection." The term "open" was chosen to emphasize that by conforming to OSI standards, a system would be open to communication with any other system obeying the same standards anywhere in the world.

In 1978, it was clear that the commercial endeavors to exploit the emerging communication technology would wait neither for SC16 to leisurely develop communication standards nor for the research community to answer most of the outstanding questions. If there was to be a consistent set of international standards, OSI would have to lead rather than follow commercial development and make use of the most recent research work when available. The size of the task would require the work to be divided among several working groups each developing standards; however, close overall coordination would also be necessary.

The first meeting of SC16 was held in March 1978. Initial discussions revealed [3] that a consensus could be reached rapidly on a basic layered architecture which would satisfy most requirements of OSI and which could be extended later to meet new requirements. SC16 decided to give the highest priority to the development of a standard Model of Architecture which would constitute the framework for the development of standard protocols.

Manuscript received March 1, 1983; revised August 20, 1983.

J. Day is with Codex Corporation, Mansfield, MA 02046.
H. Zimmermann is with the Centre National d'Études des Télécommunications, 92131 Issy-Les-Moulineaux, France.

After less than 18 months of discussions, this task was completed, and the Reference Model of Open Systems Interconnection, was transmitted by SC16 to TC97 along with recommendations to start a number of projects for developing an initial set of standard protocols for OSI. These recommendations were adopted by TC97 at the end of 1979 as the basis for development of standards for Open Systems Interconnection within the ISO. The OSI Reference Model was also recognized by the CCITT Rapporteur's Group on Public Data Network Services. At this time, SC16 began development of standard OSI Protocols for the upper four layers. These are discussed in more detail in subsequent articles in this special issue.

In late 1980, SC16 recommended that the Reference Model be forwarded as a Draft Proposal (DP) for an International Standard. After two rounds of comments, the Reference Model was progressed as a Draft International Standard (DIS) in the Spring of 1982. Comments on this vote were processed late in 1982, and the Basic Reference Model became an International Standard (ISO 7498) [4] in the Spring of 1983.

In most cases, the job of a standards committee is to take sets of commercial practices and the current research results when applicable and codify these procedures into a single standard that can be utilized by commercial products. SC16 was presented with a somewhat different problem: develop a set of standards which emerging products could converge to before the commercial practices were in place and while many of the more fundamental research problems remained unsolved. It would be presumptious to say that SC16 solved this problem. They did, however, find a way to cope with the problem in such a way as to maximize flexibility and to minimize the impact of change brought on by new technologies or new techniques.

The approach adopted by SC16 was to use a layered architecture to break up the problem into manageable pieces. The OSI Reference Model is a framework for coordinating the development of OSI standards. In OSI, the problem is approached in a top-down fashion, starting with a description at a high level of abstraction which imposes few constraints, and proceeding to more and more refined descriptions with tighter and tighter constraints. In the world of OSI, three levels of abstractions are explicitly recognized: the architecture, the service specifications, and the protocol specifications (see Fig. 1).

The OSI Architecture is the highest level of abstraction in the OSI scheme. The term "architecture" can be a very tricky term. It has been used to describe everything from a framework for development, to a particular form of organization, to hardware. A good way to think about the term is to consider the difference between an architecture and a building built to that architecture. For example, Victorian architecture is a set of rules and stylistic conventions that characterize a particular form. A Victorian building is a building built to those rules and conventions. You cannot walk into a Victorian architecture; you can walk into a

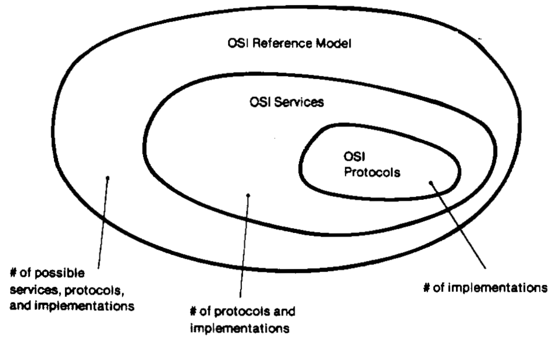

OSI Reference Model
OSI Services
OSI Protocols
# of possible services, protocols, and implementations
# of protocols and implementations
# of implementations

Fig. 1. The OSI Reference Model, Services, and Protocols are successively more detailed and therefore more constraining specifications.

Victorian building. In computer science we often mistakenly refer to a building as an architecture. More formally this can also be considered as the distinction between the type of an object (architecture) and an instance of that object (building). The OSI Reference Model defines types of objects that are used to describe an open system, the general relations among these types of objects, and the general constraints on these types of objects and relations. Specifications for the lower levels of abstractions may define other relations and tighter constraints for their purposes, but these must be consistent with those defined in the Reference Model.

The document which describes the OSI Architecture, ISO 7498, defines these objects, relations, and constraints, and also defines a seven-layer model for interprocess communication constructed from these objects, relations, and constraints. These are used as a framework for coordinating the development of layer standards by OSI committees, as well as the development of standards built on top of OSI.

The OSI Service Specifications represent a lower level of abstraction that define in greater detail the service provided by each layer. This concept is defined in greater detail in the article by Linington in this issue. The service specification will define tighter constraints than the Reference Model on the protocols and implementations that will satisfy the requirements of the layer. A service specification defines the facilities provided to the user of the service independent of the mechanisms used to accomplish the service. It also defines an abstract interface for the layer, in the sense that it defines the primitives that a user of the layer may request with no implication of how or if that interface is implemented.

The OSI Protocol Specifications represent the lowest level of abstraction in the OSI standards scheme. Each protocol specification defines precisely what control information is to be sent and what procedures are to be used to interpret this control information. The protocol specifications represent the tightest constraints placed on implementations built to conform to OSI standards.

As shown in Fig. 1, the three levels of abstraction used by OSI define successively tighter constraints on what will satisfy OSI. There are many services and protocols that satisfy the constraints required by the Reference Model. There are fewer protocols that satisfy both the Reference Model and the OSI Service Specifications. Finally, the Protocol Specifications constrain implementations sufficiently to allow open systems to communicate while still allowing differences in implementations.

Products can satisfy the much weaker constraints imposed by the Reference Model, but may not be able to communicate with open systems unless they also conform the OSI services and protocols. The OSI Reference Model cannot be implemented, and it does not represent a preferred implementation approach. It is a model for describing the concepts for coordinating the parallel development of interprocess communication standards. One must remember that in the world of OSI, only OSI protocols can be implemented and products can only conform to OSI protocols. Thus the statement "this product conforms to the OSI Reference Model" does not imply the ability to interwork with other products which may make the same claim.

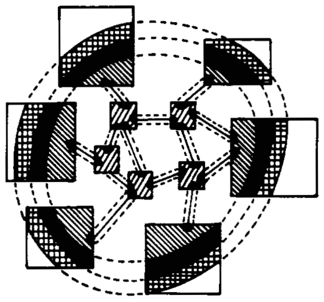


Fig. 2. Network layering.

The purpose of OSI is to allow any computer anywhere in the world to communicate with any other, as long as both obey the OSI standards. OSI standards and the degree of compatibility required to meet this goal make formal description methods a necessity. SC16/WG1 on Architecture established a group early in its work to develop formal description methods for defining the protocols so that they could be implemented unambiguously by people all over the world without having to consult with a few experts on how to interpret the standard. The article in this issue by Vissers, Tenney, and Bochmann gives more details on the OSI formal description methods as used by ISO.

In the remainder of this paper we describe the basic concepts used in the Reference Model, then give a brief description of each of the seven layers and identify a few of the outstanding architectural issues.

II. The Elements of the Architecture

ISO 7498, the document describing the basic OSI Reference Model, is divided into two major sections. The first of these describes the elements of the architecture. These constitute the building blocks that are used to construct the seven-layer model. The second describes the services and functions of the layers.

A. Systems, Layer, and Entities

The OSI Reference Model is an abstract description of interprocess communication. OSI is concerned with standards for communication between systems. In the OSI Reference Model, communication takes place between application processes running in distinct systems. A system is considered to be one or more autonomous computers and their associated software, peripherals, and users that are capable of information processing and/or transfer. Although OSI techniques could be used within a system (and it would be desirable for intra- and inter-system communication to appear as similar as possible to the user), it is not the intent of OSI to standardize the internal operation of a system.

Layering is used as a structuring technique to allow the network of open systems to be logically decomposed in independent, smaller subsystems (see Fig. 2). Each individual system itself is

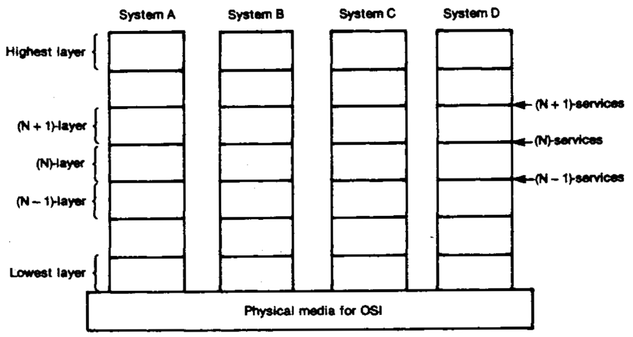


Fig. 3. Systems, layers, and services in the OSI environment.

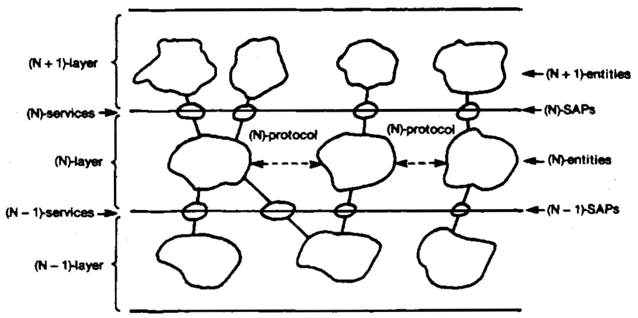


Fig. 4. Entities, service access points (SAP's), and protocols.

viewed as being logically composed of a succession of subsystems, each subsystem corresponding to the intersection of the system with a layer. In other words, a layer is viewed as being locally composed of subsystems of the same rank in all interconnected systems. Each subsystem, in turn, is viewed as being made of one or several entities. A layer, therefore, comprises many entities distributed among interconnected open systems. Entities in the same layer are termed peer entities.

For simplicity, any layer is referred to as the (N)-layer, while its next lower and next higher layers are referred to as the $(N-1)$-layer and the $(N+1)$-layer, respectively. The same notation is used to designate all concepts relating to layers, e.g., entities in the (N)-layer are termed (N)-entities, and illustrated in Figs. 3 and 4.

The basic idea of layering is that each layer adds value to services provided by the set of lower layers in such a way that the highest layer is offered the full set of services needed to run distributed applications. Layering thus divides the total problem into smaller pieces.

Another basic principle of layering is to ensure independence of each layer by defining services provided by a layer to the next higher layer, independent of how these services are performed. This permits changes to be made in the way a layer or a set of layers operate, provided they still offer the same service to the next higher layer. This technique is similar to the one used in structured programming where only the functions performed by a module (and not its internal functioning) are known by its users.

B. *Services and Service Access Points*

Each layer provides services to the layer above (with the exception of the highest layer). A service is a capability of the (N)-layer which is provided to the $(N+1)$-entities. But it is important to note that not all functions performed within the (N)-layer are services. Only those capabilities that can be seen from the layer above are services.

(N)-entities distributed among the interconnected open systems work collectively to provide the (N)-service to $(N+1)$-entities as illustrated in Fig. 4. In other words, the (N)-entities add value to the $(N-1)$ service they get from the $(N-1)$-layer and offer this value-added service, i.e., the (N)-service to the $(N+1)$-entities.

The (N)-services are offered to the $(N+1)$-entities at the (N)-service access points, or (N)-SAP's for short, which represent the logical interfaces between the (N)-entities and the $(N+1)$-entities. An $(N+1)$-entity communicates with an (N)-entity in the same system through an (N)-SAP. An (N)-SAP can be served by only one (N)-entity and used by only one $(N+1)$-entity, but one (N)-entity can serve several (N)-SAP's and one $(N+1)$-entity can use several (N)-SAP's. An (N)-SAP is located by its (N)-address (see Section II-D).

C. *Functions and Protocols*

An (N)-function is part of the activity of an (N)-entity. Flow control, sequencing, data transformation are all examples of (N)-functions. Cooperation among (N)-entities is governed by one or more (N)-protocols. An (N)-protocol is the set of rules and formats which govern the communication between (N)-entities performing the (N)-functions in different open systems. In particular, direct communication between the (N)-entities in the same system, e.g., for sharing resources, is not visible from outside the system and thus is not covered by the OSI Architecture.

D. *Naming*

Objects within a layer or at the boundary between adjacent layers need to be uniquely identifiable, e.g., in order to establish a connection between two SAP's, one must be able to identify them uniquely. The OSI Architecture defines identifiers for entities, SAP's, and connections as well as relations between these identifiers, as briefly outlined below.

Each (N)-entity is identified with a global title which is unique and identifies the same (N)-entity anywhere in the network of open systems. Within more limited domains, an (N)-entity can be identified with a local title which uniquely identifies the (N)-entity only in that domain. For instance, within the domain corresponding to the (N)-layer, (N)-entities are identified with (N)-global titles which are unique within the (N)-layer.

Each (N)-SAP is identified by an (N)-address which uniquely locates the (N)-SAP at the boundary between the (N)-layer and the $(N+1)$-layer. The concepts of titles and addresses are illustrated in Fig. 6.

Bindings between (N)-entities and the $(N-1)$-SAP's they use (i.e., SAP's through which they can access each other and communicate) are defined in an (N)-directory which indicates correspondence between global titles of (N)-entities and (N)-addresses through which they can be reached.

Correspondence between (N)-addresses served by an (N)-entity and the $(N-1)$-addresses used for this purpose is performed by an (N)-mapping function. In addition to the simplest case of one-to-one mapping, mapping may, in particular, be hierarchical with the (N)-address being made of an $(N-1)$-address and an (N)-suffix. Mapping may also be performed "by table lookup."

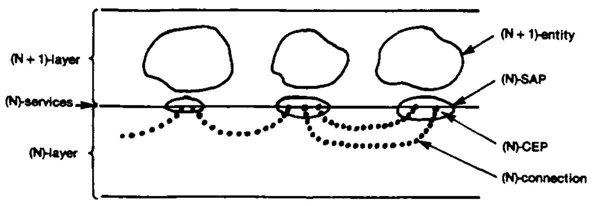


Fig. 5. Connections and connection endpoints (CEP's).

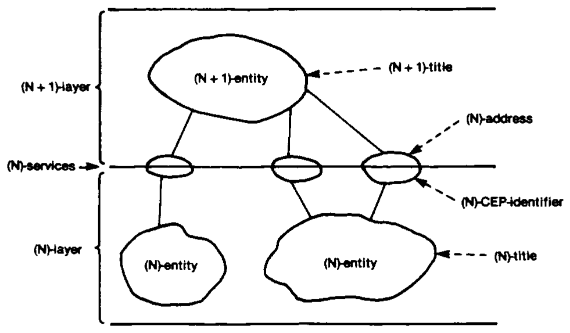
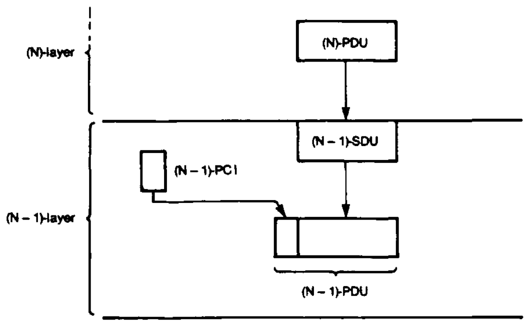


Fig. 6. Titles, addresses, and CEP identifiers.

	Control	Data	Combined
(N) - (N) Peer Entities	(N)-Protocol-Control-Information	(N)-User-Data	(N)-Protocol-Data Units
(N)-(N − 1) Adjacent layers	(N − 1)-Interface-Control-Information	(N − 1) Interface Data	(N − 1)-Interface Data-Unit

Fig. 7. Interrelationship between data units.



PCI = Protocol-control-information
PDU = Protocol-data-unit
SDU = Service-data-unit

Fig. 8. Logical relationship between data units in adjacent layers.

As work has progressed in OSI, two distinctions have come to be recognized as critical to the naming problem. First is the recognition that one must distinguish and be able to name separately and uniquely types and instances. Second, there are two types of names: primitive and descriptive. Primitive names are unique and assigned by some domain administrator, e.g., phone numbers, social security numbers, etc. Descriptive names are composites of primitive names, keywords, etc., that can be resolved to primitive names by interpretation, e.g., my name and address. Relative few types and instances require primitive names. Descriptive names for everything else can be built from these as well as synonyms for objects with primitive names. This greatly simplifies the administration of naming in OSI.

E. Connections

A common service offered by all layers consists of providing associations between peer SAP's which can be used in particular to transfer data (as well as for other purposes such as to synchronize the served entities participating in the association). More precisely (see Fig. 5), the (N)-layer offers (N)-connections between (N)-SAP's as part of the (N)-services. The most usual type of connection is the point-to-point connection, but there are also multi-endpoint connections which correspond to multiple associations between entities (e.g., broadcast or multidrop communications). The end of an (N)-connection at an (N)-SAP is called an (N)-connection-endpoint or (N)-CEP for short. Several connections may coexist between the same pair (or n-tuple) of SAP's.

Each (N)-CEP is uniquely identified within its (N)-SAP by an (N)-CEP identifier which is used by the (N)-entity and the $(N+1)$-entity on both sides of the (N)-SAP to identify the (N)-connection, as illustrated in Fig. 6. This is necessary since several (N)-connections may end at the same (N)-SAP.

The Basic Reference Model currently restricts communications between (N)-entities to "connection mode." In this mode, the $(N-1)$-service requires that an $(N-1)$-connection be established between $(N-1)$-SAP's before any communication between (N)-entities can take place. Conversely, when the (N)-entities need no longer communicate, the $(N-1)$-connection can be released. This connection mode covers traditional teleprocessing. For newer applications, a "connectionless" mode is currently being developed within ISO as a complement to the connection mode.

1) Establishment and Release of Connections: When an $(N+1)$-entity requests the establishment of an (N)-connection from one of the (N)-SAP's it uses to another (N)-SAP, it must provide at the local (N)-SAP the (N)-address of the distant (N)-SAP. When the (N)-connection is established, both the $(N+1)$-entity and the (N)-entity will use the (N)-CEP identifier to designate the (N)-connection.

(N)-connections may be established and released dynamically on top of $(N-1)$-connections. Establishment of an (N)-connection implies the availability of an $(N-1)$-connection between the two entities. If not available, the $(N-1)$-connection must be established. This requires the availability of an $(N-2)$-connection. The same consideration applies downwards until an available connection is encountered.

In some cases, the (N)-connection may be established simultaneously with its supporting $(N-1)$-connection provided the $(N-1)$-connection establishment service permits (N)-entities to exchange information necessary to establish the (N)-connection.

2) Data Transfer on a Connection: Information is transferred in various types of data units between peer entities and between entities attached to a specific service access point. The data units are defined below and the interrelationship among several of them is illustrated in Figs. 7 and 8.

(N)-protocol control information is information exchanged between two (N)-entities, using an $(N-1)$-connection, to coordinate their joint operation.

(N)-user-data are the data transferred between two (N)-entities on behalf of the $(N+1)$-entities for whom the (N)-entities are providing services.

An (N)-protocol-data-unit is a unit of data which contains (N)-protocol-control-information and possibly (N)-user-data.

(N)-interface-control-information is information exchanged between an $(N+1)$-entity and an (N)-entity to coordinate their joint operation.

(N)-interface-data-unit is the amount of $(N-1)$-interface-data whose identity is preserved from one end of an $(N-1)$-connection to the other. Data may be held within a connection until a complete service data unit is put into the connection.

Expedited (N)-service-data-unit is a small (N)-service-data-unit whose transfer is expedited. The $(N-1)$-layer ensures that an expedited data-unit will not be delivered before any subsequent service-data-unit or expedited data-unit sent on that connection. An expedited (N)-service-data-unit may also be referred to as an (N)-expedited-data-unit.

Note: An (N)-protocol-data-unit may be mapped one-to-one onto an $(N-1)$-service-data-unit (see Fig. 8).

3) Elements of Layer Operation: There are a number of functions which are recognized as part of layer operation. These include such things as multiplexing, flow control, and error control. As the Reference Model matures other elements will be added. In the Reference Model, these elements are described in general without reference to a particular layer.

Three particular types of construction of (N)-connections on top of $(N-1)$-connections are distinguished.

a) One-to-one correspondence, where each (N)-connection is built on one $(N-1)$-connection.

b) Multiplexing, where several (N)-connections are multiplexed on one single $(N-1)$-connection.

c) Splitting, where one single (N)-connection is built on top of several $(N-1)$-connections, the traffic on the (N)-connection being divided between the various $(N-1)$-connections.

Two forms of flow control are recognized by the reference model: a peer flow control which regulates the flow of (N)-protocol-data-units between entities within the same layer, and interface flow control which regulates the flow of (N)-interface-data between an $(N+1)$-entity and (N)-entity through an (N)-SAP.

A variety of error functions are recognized by the model including acknowledgment, error detection, and error notification mechanisms. The model also describes a reset function to allow recovery from a loss of synchronization between communicating (N)-entities.

III. The Seven-Layer Model

In the last section, the basic elements of the OSI Reference Model were developed. These serve as the building blocks for constructing the model of interprocess communication. In OSI, interprocess communication is subdivided into seven independent layers. Each (N)-layer uses the services of the lower $(N-1)$-layer and adds the functionality peculiar to the (N)-layer to provide service to the $(N+1)$-layer above. Layers have been chosen to break up the problem into reasonably sized smaller problems that can be considered relatively independently. The seven layers are described briefly below (see Fig. 9).

A. Application Layer

The Application Layer as the highest layer of OSI does not provide services to any other layer. The primary concern of the Application Layer is with the semantics of the application. All

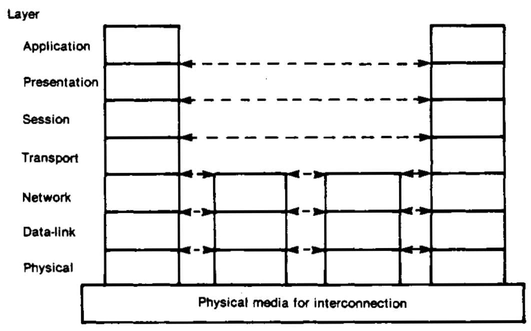


Fig. 9. The seven-layer OSI architecture.

application processes reside in the Application Layer. However, only part of the Application Layer is in the real OSI system. Those aspects of the application process concerned with interprocess communication (called the application entity) are within the OSI environment. SC16 is developing the Common Application Service Elements that provide common procedures for constructing application protocols and for accessing the services of OSI. SC16 is also developing three application protocols of general interest (virtual file, virtual terminal, and job transfer and manipulation services), as well as OSI application and system management protocols. However, the bulk of application protocols will be defined by the users of OSI. The common Application Service is the only means by which users of OSI access OSI services.

B. Presentation Layer

The primary purpose of the Presentation Layer is to provide independence to application processes from differences in data representation, i.e., syntax. The Presentation Layer protocol allows the user to select a "Presentation Context." The Presentation Context may be specific to an application such as a library protocol or virtual terminal, to a type of hardware such as a particular machine representation or to some standard or canonical representation. Thus a user of OSI wanting to develop an OSI application protocol defines an application protocol using the relevant parts of the Common Application Service Elements and a Presentation Context which defines the representation of the data to be transferred. The OSI user may use an existing context or define his own and register it with ISO. Draft Proposed Standards for Presentation Layer services and protocols are expected in early 1984.

C. Session Layer

The primary purpose of the Session Layer is to provide the mechanisms for organizing and structuring the interactions between application processes. The mechanisms provided in the Session Layer allow for two-way simultaneous and two-way alternate operation, the establishment of major and minor synchronization points, and the definition of special tokens for structuring exchanges. In essence, the Session Layer provides the structure for controlling the communication. The OSI Session Service and Protocol are now being processed as Draft Proposed (DP) Standards.

D. Transport Layer

The purpose of the Transport Layer is to provide transparent transfer of data between end systems, thus relieving the upper

DOCKET ALARM

Explore Litigation Insights

Docket Alarm provides insights to develop a more informed litigation strategy and the peace of mind of knowing you're on top of things.

Real-Time Litigation Alerts

Keep your litigation team up-to-date with **real-time alerts** and advanced team management tools built for the enterprise, all while greatly reducing PACER spend.

Our comprehensive service means we can handle Federal, State, and Administrative courts across the country.

Advanced Docket Research

With over 230 million records, Docket Alarm's cloud-native docket research platform finds what other services can't. Coverage includes Federal, State, plus PTAB, TTAB, ITC and NLRB decisions, all in one place.

Identify arguments that have been successful in the past with full text, pinpoint searching. Link to case law cited within any court document via Fastcase.

Analytics At Your Fingertips

Learn what happened the last time a particular judge, opposing counsel or company faced cases similar to yours.

Advanced out-of-the-box PTAB and TTAB analytics are always at your fingertips.

API

Docket Alarm offers a powerful API (application programming interface) to developers that want to integrate case filings into their apps.

LAW FIRMS
Build custom dashboards for your attorneys and clients with live data direct from the court.

Automate many repetitive legal tasks like conflict checks, document management, and marketing.

FINANCIAL INSTITUTIONS
Litigation and bankruptcy checks for companies and debtors.

E-DISCOVERY AND LEGAL VENDORS
Sync your system to PACER to automate legal marketing.

WHAT WILL YOU BUILD? | sales@docketalarm.com | 1-866-77-FASTCASE

fastcase® Smarter legal research.