The ISO Network Reference Model

Like all models, the ISO Model* helps to clarify thought—something sorely needed in computer networking, where the complexities can easily exceed human comprehension.

by Capt. William T. Brewer

Is your computer system an "introvert" or an "extrovert"? That probably sounds like a strange question; but in a sense, computers can be characterized in this way. In the centralized systems of the past (Figure 1a), most computers were "introverts," their operating systems being incapable of "talking" to operating systems in other computers. As a result, their entire existence was spent in isolation, not knowing or caring that other systems existed.

But with today's trend toward distributed systems (Figure 1b), computers lead a much more "extroverted" lifestyle, conversing routinely with multiple computers often separated by thousands of miles. This blossoming from "introversion" to "extroversion" has brought with it a new type of system software often described as a "distributed operating system."

Essentially, a distributed operating system allows a group of computers to operate as a single, "virtual" computer with great potential for sharing software, hardware, and data with more reliability and flexibility. But there is a price to pay for all of these attractive qualities. By "distributing" the operating system across multiple computers, we also "distribute" the complexity of the overall system across a wider variety and larger number of users and support personnel. In a centralized system, only a small number of people are concerned with the system software, which usually belongs to a single vendor. But in a distributed system, a much larger number of people are involved with multiple

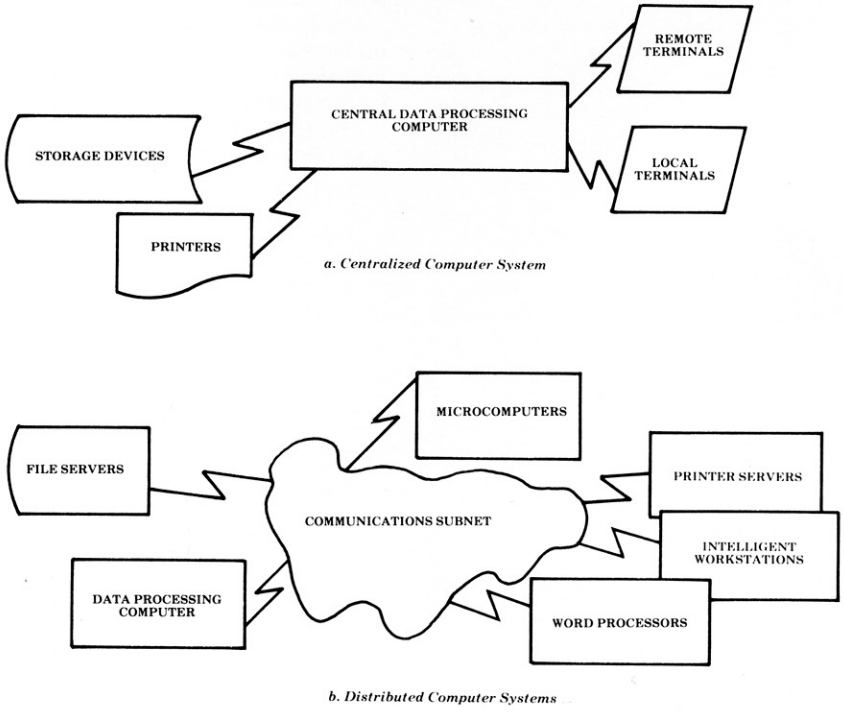


Figure 1.

To help readers with the difficult terminology of this article, a list of ISO definitions appears on page 16.

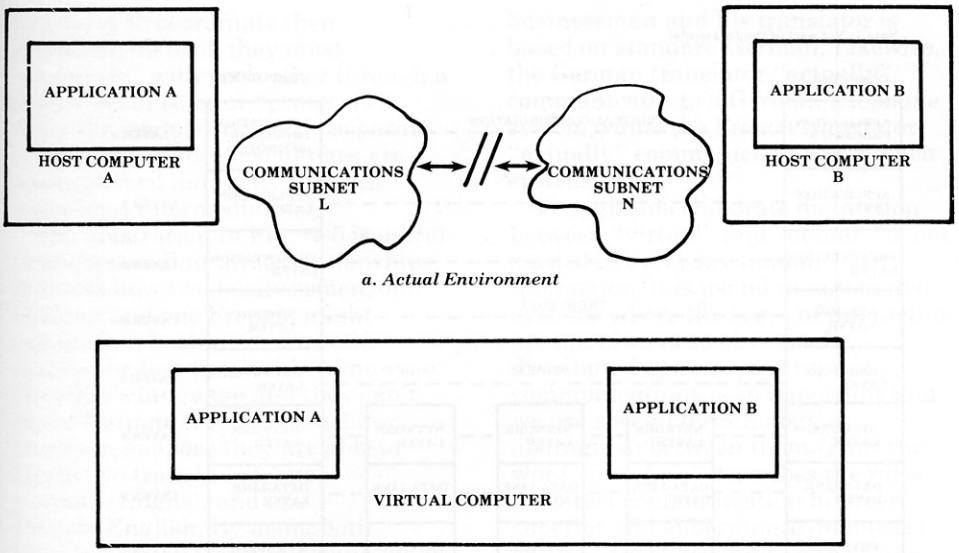


Figure 2.

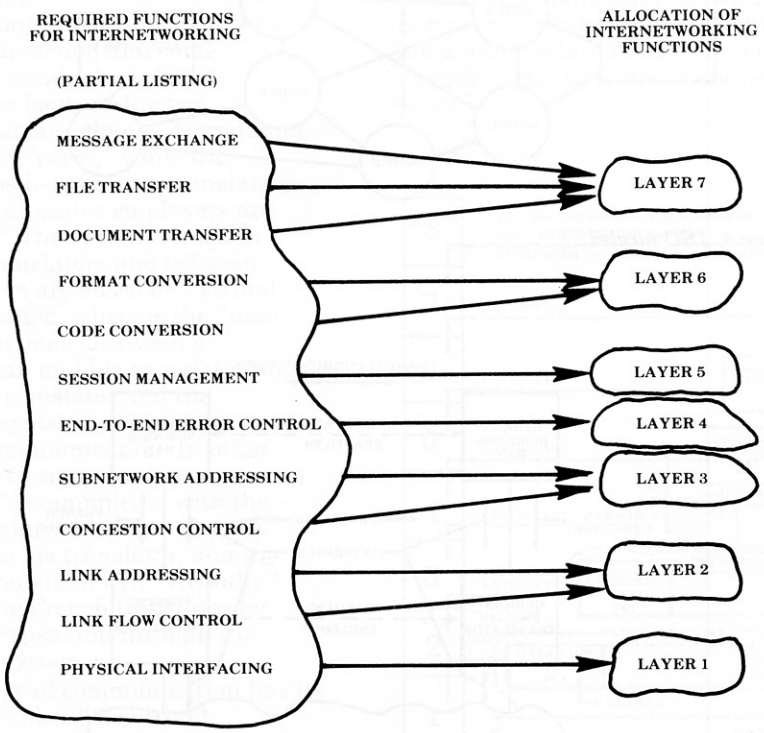


Figure 3. Internetworking functions

computer systems from multiple vendors connected by multiple communications suppliers. Thus, a distributed system requires a broader understanding of the overall system architecture by a larger number of people and a greater standardization of equipment and procedures.

The International Standards Organization (ISO) is attempting to fill both of these requirements. In 1977, ISO began development of a standard reference model for the internetworking of computer systems. The result of that work, the Network Reference Model for Open Systems Interconnection (OSI), describes a distributed architecture that will allow processes running on different computers separated by multiple communications subnets to communicate with each other as if the two processes were running on the same machine (Figure 2).

Like all models, the ISO Model helps to clarify thought—something sorely needed in computer networking, where the complexities can easily exceed human comprehension. The architecture described by the ISO Model reduces the complexity involved in computer-to-computer communications by grouping the large number of required functions into layers (Figure 3). This layering simplifies system design by structuring the network architecture into smaller, more manageable elements in much the same way that modular programming has simplified the design, programming, maintenance, and modification of software in general.

The modular concept inherent in the ISO Model provides an excellent framework for the development of standards for each layer. But in addition to promoting this type of standardization, the model also is encouraging a wider and better understanding of distributed systems by standardizing the terminology associated with these systems. In fact, being able to "talk ISO" is already the mark that separates the computer-ignorant from the computer-informed at all levels. The growing importance of the Defense Data Network (DDN) should make military communicators especially eager to learn about the ISO Model and its associated terminology.

An overview of the ISO Model

Most people are more accustomed to viewing a network from the standpoint of its "physical architecture" or "topology" shown in the botton portion of Figure 4, rather than its "functional architecture," shown in the top of Figure 4. Both viewpoiints are valid, but the functional perspective is more useful in conveying the actual operation of a distributed system; consequently, this article will emphasize the functional viewpoint and associated "protocols."

"Protocols," rules whereby entities interact, are essential in distributed systems to coordinate the system functions that have been distributed. Though many centralized systems do exhibit some degree of functional distribution (such as distribution of input and output functions), protocols necessarily play a larger role in distributed systems than in centralized systems because distributed systems naturally have a larger number of distributed functions.

Protocol entities can be implemented in a variety of ways. In distributed systems that are automated, most of these entities are based in sofware modules, with a smaller number being based in firmware and hardware. However, the type of implementation (hardware, software, firmware, or even manual) is more important to the speed and accuracy of a system than it is to the functional architecture. For example, the functional architecture of a distributed system would not necessarily change as a result of automating existing manual procedures or moving software-based functions to firmware. This article emphasizes functional architecture because it allows us to focus on *what* needs to be accomplished without worrying about the numerous details of *how* it will be done.

While functions in a centralized system are usually allocated to a single "entity," functions in a distributed system are usually allocated to several modules residing on widely separated computers. For example, the error control function performed across a link or across a network is performed by multiple modules working in coordination with each other even though they may be separated by hundreds or even thousands of miles. The modules hosting a "distributed function" don't have centralized control or direct

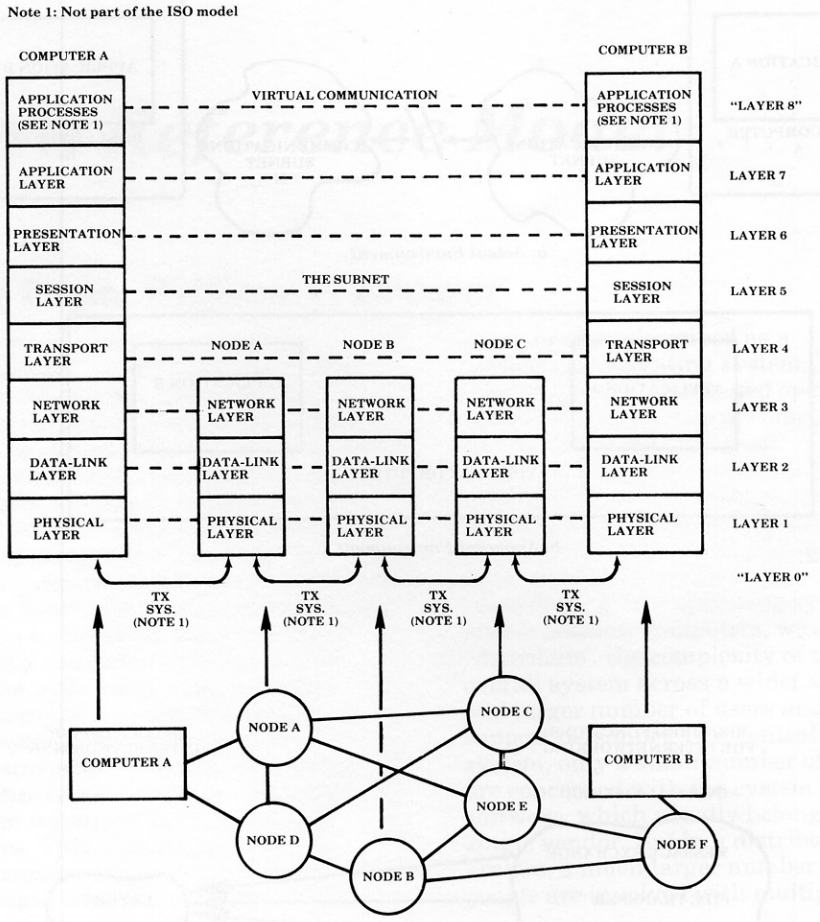


Figure 4. ISO model

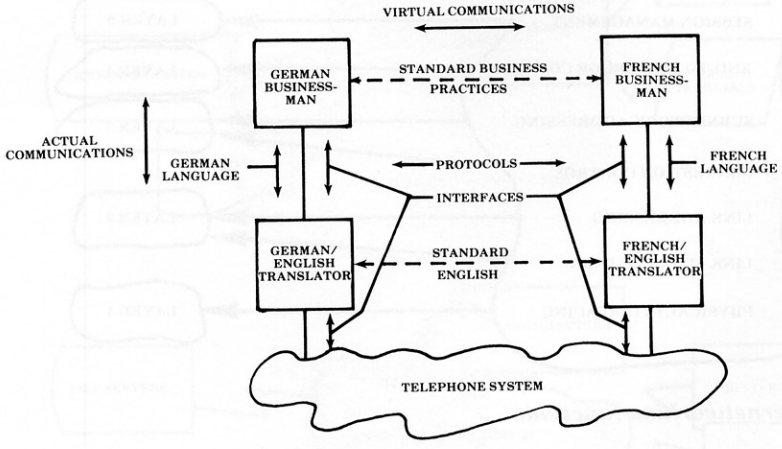


Figure 5. "Layered architecture"

interfaces to coordinate their operation; instead, they must "cooperate" with each other through a precise set of rules or "protocols." Since the modules reside on separate computers, protocol exchanges are accomplished indirectly through lower-level intermediaries.

The illustration in Figure 5 is useful in understanding these relationships. It shows how two businessmen, one German and one French, might engage in a business transaction despite the fact that neither knows the other's language and they can't find a German/French translator. However, suppose they are able to locate two translators, one German/English and one French/English. By using both translators, the German businessman is able to communicate with the French businessman by speaking to the German/English translator, who in turn speaks in English to the French/English translator, who in turn speaks French to the French businessman.

This arrangement illustrates two types of interaction and some associated terminology. The interactions between the two businessmen and the two translators are between "peers," while the interactions between the translators and their respective employers are "non-peer." The "peer interaction" between translators and between businessmen are based on "virtual" communication, whereas the "non-peer interactions" between a businessman and his translator and between a translator and the telephone system are based on "actual" communication. In other words, the German businessman can "virtually" communicate with the French businessman by "actually" speaking to his translator, and the German translator can "virtually" speak to the French translator by "actually" speaking through the telephone system.

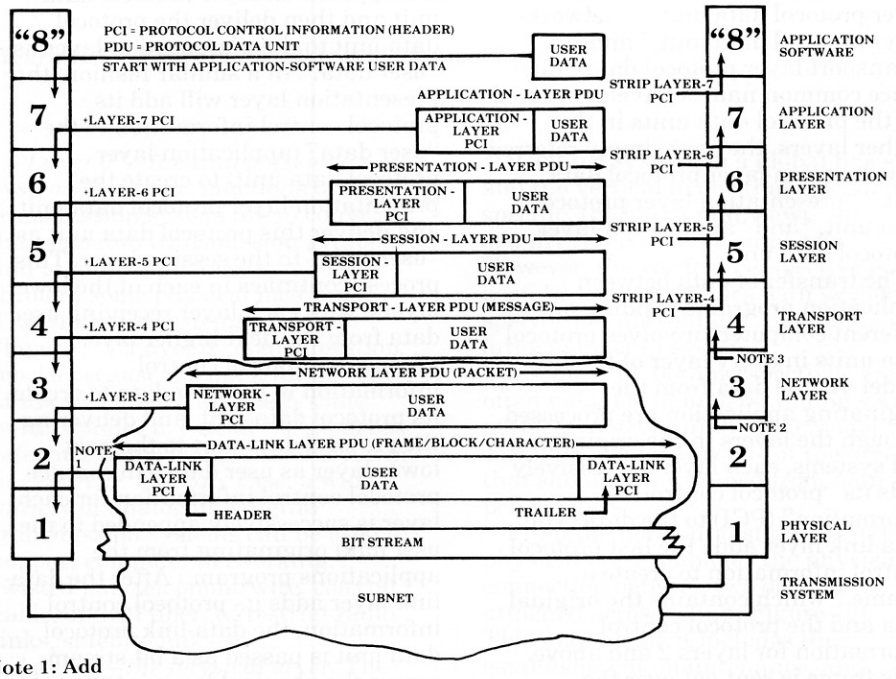
Each type of communication has its own set of rules. The virtual communication between the businessmen is based on standard business practices, and the virtual communication between translators is based on standard English. On the other hand, the actual communication between the French businessman and his translator is based on standard French, while the actual communication between the German businessman and his translator is based on standard German. Likewise, the German translator "actually" communicates to a German telephone system, while the French translator "actually" communicates to a French system.

Though this elaborate distinction between "virtual" and "actual " is not needed in everyday human interaction, it is useful in automated systems where the types of interaction are not always so obvious. In distributed systems, the type of communications is so important that we use special terminology to distinguish between them. Thus the word "interface" describes the rules for direct communication between superior and subordinate (non-peer) entities (those residing on the same computer) while the word "protocol" describes the rules for indirect communication between peer-level entities (those residing on different computers). The former defines a "vertical" relationship based on "actual" communication between "non-peers," while the latter defines a "horizontal" relationship based on "virtual" communication between "peers."

This distinction between "interfaces" and "protocols" is useful because interfaces need not be standardized across all end systems in a distributed system, but protocols must be standardized. Interfaces don't require standardization because they are specific to a particular end system; but protocols aren't system-specific and must be standardized since they must be obeyed by all end systems. This means an end system can operate perfectly well without knowing anything about the interfaces in other end systems as long as it knows everything about the protocols used between end systems. One important aspect of these protocols is the associated "protocol data units," which are the basis for "protocol exchanges" between end systems.

Protocol data units (PDUs)

Do you know the difference between a "frame" and a "packet"? Or a "block" and a "message"? If you do, you should also know that these words are often used loosely even though they have specific meanings and relationships which should be respected. For example, each of these



Note 1: Add Layer-2 PCI

Note 2: Strip Layer-2 PCI

Note 3: Strip Layer-2 PCI

Figure 6. Protocol data units

terms identifies a specific "protocol data unit" in a layered protocol architecture. These protocol data units are related to each other; thus a "message" (or part of a message) is contained in a "packet," and a "packet" is contained inside a "frame." Also, a "block" and a "frame" are essentially the same except that the former term is used with respect to message-switching networks, while the latter term is used with respect to other types of networks. All of this is confusing unless you have a basic understanding of typical protocol data units found in a distributed system.

"Protocol data units" (Figure 6) are inherent in distributed systems; and as previously noted, the term includes the familiar "frame," "block," "packet," and "message." But the term is more inclusive than this and applies to data units in every layer of the ISO Reference Model. For example, the protocol data unit for the physical layer is the "bit stream"; for the data-link layer, it's the "frame" or "block"; for the network layer, it's the "packet"; and, for the transport layer, it's the "message." Actually, the terms "frame," "block," "packet," and "message" are common names; the more formal names are "data-link layer protocol data unit," "network-layer protocol data unit," and "transport-layer protocol data unit." Since common names haven't evolved for the protocol data units in the higher layers, they are simply referred to as "session-layer protocol data unit," "presentation-layer protocol data unit," and "application-layer protocol data unit."

The transfer of data between application programs residing on different computers involves protocol data units in every layer of the ISO Model. As the data from the originating application are processed through the layers in the originating end systems, each layer successively adds its "protocol control information" (PCI) to the data. The data-link layer adds the last protocol control information to create a "frame," which contains the original data and the protocol control information for layers 2 and above. This frame is sent out over the intervening subnet and eventually arrives at the destination end system. At the destination end system, each layer processes the received data according to its protocol control information and then "strips" its protocol control information before delivery to the next higher layer. The application layer is the last layer to strip its protocol control information from its protocol data unit; when it does, the remaining user data represents the data sent from the applications program on the originating end system.

The assembly and disassembly of protocol data units in the originating and destination end systems is a very systematic process. In general, the "n-layer" (the layer under consideration) protocol data unit is created by adding the n-layer protocol control information to the n-layer "user data." The n-layer user data and the n-1 protocol data unit are essentially the same. As you can see, the terms "user data," "protocol control information," and "protocol data unit" are relative: therefore, they can be ambiguous unless the layer under consideration is specified.

Every layer in the ISO Model has its own user data, protocol control information, and protocol data unit. For example, an applications program delivers "user data" to the local application layer. The applications layer will add its protocol control information to the user data to create the application-layer protocol data unit and then deliver the protocol data unit to the presentation layer as "user data." In a similar fashion, the presentation layer will add its protocol control information to the "user data" (application-layer protocol data unit) to create the presentation-layer protocol data unit and deliver this protocol data unit as "user data" to the session layer. This process continues in each of the lower layers, with each layer receiving user data from the next higher layer, adding its protocol control information to the user data to create its protocol data unit, and delivering the protocol data unit to the next lower layer as user data. (In essence, protocol control information for each layer is successively appended to the user data originating from the applications program.) After the data-link layer adds its protocol control information, the data-link protocol data unit is passed as a bit stream across the physical-layer interface to the transmission system and over to the destination end system.

Disassembly of protocol data units at the destination end system is essentially the reverse of the preceding process. At the destination end system, each layer in succession examines the contents of its respective protocol control information, strips its protocol control information from its protocol data unit, and delivers the remainder as "user data" to the next higher layer. By the time the application layer at the destination end system has stripped its protocol control information from its protocol data unit, all that's left are the original user data, which are delivered to the destination applications program.

The purpose of all this "overhead" is to make the transfer of data between remote applications as easy as if the applications were residing on the same computer. Ideally, the distributed nature of the system should be totally hidden, or "transparent," to the communicating applications.

Transparency

The proper operation of a complex distributed system depends greatly on the concept of "transparency." Think of "transparent" as the opposite of "virtual": transparent means "something exists but appears not to," while virtual means "something doesn't exist but appears to."

Actually the term "transparent" applies to a layered architecture in at least two ways.

First, "transparent" aptly describes the relationship between layers in a well-designed, distributed system because, in such a system, a layer is not aware of the protocol data units in other layers. In a broader sense, all the details of how each layer operates are hidden, or transparent, to all other layers. The benefit of this type of transparency is that the protocols used in each layer can be modified or substituted with no effect on other layers as long as the interfaces between layers remain the same.

Another aspect of transparency is "data transparency," which means that the data content of a particular protocol data unit is not seen and is thus "transparent" to the protocol entities associated with the protocol data unit. A system with data transparency is not affected by arbitrary bit patterns occurring in the data exchanged between end systems; on the other hand, a system without data transparency can "crash" or "lock up" if certain bit patterns occur in the data. Lack of data transparency may not be a serious

problem if offending bit patterns can be easily restricted from appearing in the data; but if data transfers include binary files, restrictions on data content are difficult to enforce. In all cases, lack of data transparency will present complications that can be avoided through proper design.

Data transparency is achieved by following three principles. First, the protocol entities in each layer must operate according to their own dedicated protocol control information; second, the protocol entities in each layer must know nothing about the protocol control information for other layers; and third, the protocol entities in each layer must know where to find their own protocol control information. As we'll see later, the third point is trivial as long as the protocol entities know where the layer-2 frame begins and ends. But finding, or synchronizing, the layer-2 frame isn't easy, because arbitrary bit patterns in the layer-1 bit stream can be mistaken for frame delimiters, thereby disrupting the link and the supported end system.

A variety of "data transparency" techniques can be used in layer 2 to avoid "lock ups" or "crashes" caused by the data content of the layer-2 protocol data unit. These techniques include "bit stuffing," "byte/character stuffing," and "byte counts." The first two techniques operate by "disguising" any bit patterns in the data that appear to be control bits/bytes/characters by "stuffing" extra bits/bytes/characters into the data. The last technique provides data transparency by allowing the data-link entities to calculate the beginning and end of the frame using a byte or character count of the data; this eliminates the need to search for frame delimiters altogether. (The use of a fixed-length frame/block falls under this last category, since using a fixed-length frame/block represents an implied byte count.) All of these techniques ensure data transparency not only for the layer-2 frame but for higher layers as well, since the location of higher-layer protocol control information is fixed in relationship to the frame.

Now that we've covered some basic terms, let's take a quick look at the layers that make up a layered protocol architecture, beginning with transmission systems (Figure 7).

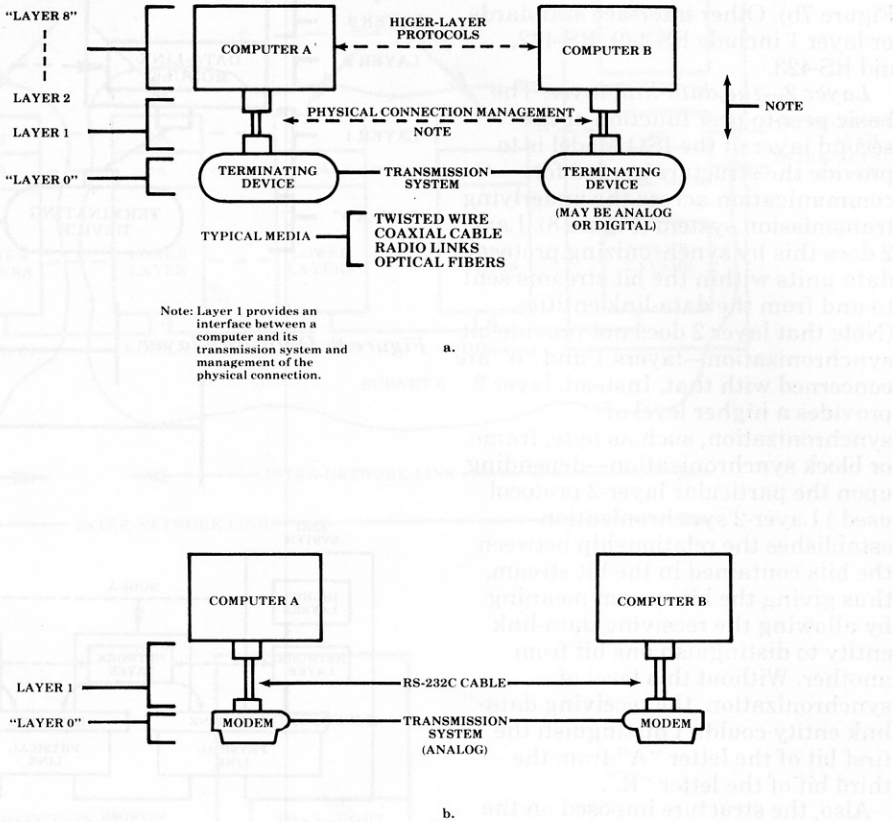


Figure 7. Transmission system and physical layer

"Layer 0"—the transmission system. The main functions of a transmission system are to convert data bits into transmission symbols and to propagate these symbols through some physical medium. Transmission systems belong to "layer 0." ("Layer 0" is in quotation marks because it's not a formal part of the ISO Model, although some writers prefer to extend the model by referring to the transmission system as "layer O.") In any case, a wide variety of analog and digital transmission systems can be used to support computer networking: twisted-pair telephone wire, coaxial cable, optical fiber, terrestrial radio links, satellite links, etc.

Layer 1—the physical layer. The physical layer, like other layers in the ISO Reference Model, can be viewed from two different perspectives, the "peer-to-peer" ("horizontal") viewpoint, and the "layer-to-layer" ("vertical") viewpoint. Because the horizontal viewpoint applies to a distributed system on a global basis and the vertical to one that is system-specific, the horizontal viewpoint usually receives more emphasis. However, the vertical perspective is useful when it coincides with system-specific details that are common to a large number of systems. Consequently, both perspectives are often valuable in understanding some of the layers within the ISO Model. The physical layer is one of the layers that should be examined from both perspectives.

From the horizontal standpoint, the physical layer manages the physical connection between two devices attached across a link (Figure 7a). This management includes the establishment, maintenance, and termination of a physical connection between end devices attached to the physical link.

From a vertical standpoint, the physical layer is essentially an interface between a computer and its

associated transmission system. A familiar example is the RS-232C cable connecting a computer to its modem (Figure 7b). Other interface standards for layer 1 include RS-449, RS-422, and RS-423.

Layer 2—the data-link layer. The basic peer-to-peer function of the second layer of the ISO Model is to provide the structure needed for communication across the underlying transmission system (Figure 8). Layer 2 does this by synchronizing protocol data units within the bit streams sent to and from the data-link entities. (Note that layer 2 does not provide bit synchronization—layers 1 and "0" are concerned with that. Instead, layer 2 provides a higher level of synchronization, such as byte, frame, or block synchronization—depending upon the particular layer-2 protocol used.) Layer-2 synchronization establishes the relationship between the bits contained in the bit stream, thus giving the bit stream meaning by allowing the receiving data-link entity to distinguish one bit from another. Without this level of synchronization, the receiving data-link entity couldn't distinguish the first bit of the letter "A" from the third bit of the letter "R."

Also, the structure imposed on the bit stream by layer 2 makes error control possible. Layer-2 error control is needed because the underlying physical link is subject to transmission impairments, causing it to be error-prone. In fact, the best known job of layer 2 is the elimination of these errors through some type of detection and correction. On asynchronous links, the detection and correction may be mostly manual with some help from simple parity checks. Synchronous links, on the other hand, use automated techniques such as cyclic redundancy checks (CRCs). In either case, layer 2 operates above layer 1 and "layer 0" to convert an error-prone physical link into an error-free data link. IBM Bisynch and Synchronous Data Link Control (SDLC) are some well-known protocols that perform this function. Military communicators may recognize Mode I, Mode II, and Mode V as layer-2 protocols used in the AUTODIN Network.

Layer 3—the network layer: The network layer encompasses those functions needed to provide a connection across multiple data links in a switched, multi-hop subnetwork. Though computers may be connected

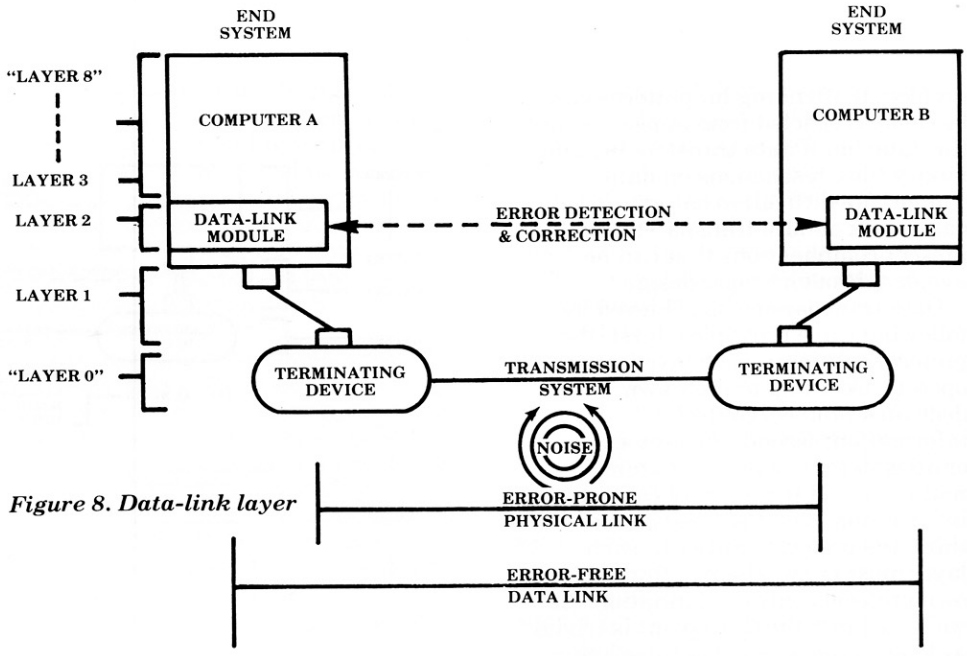


Figure 8. Data-link layer

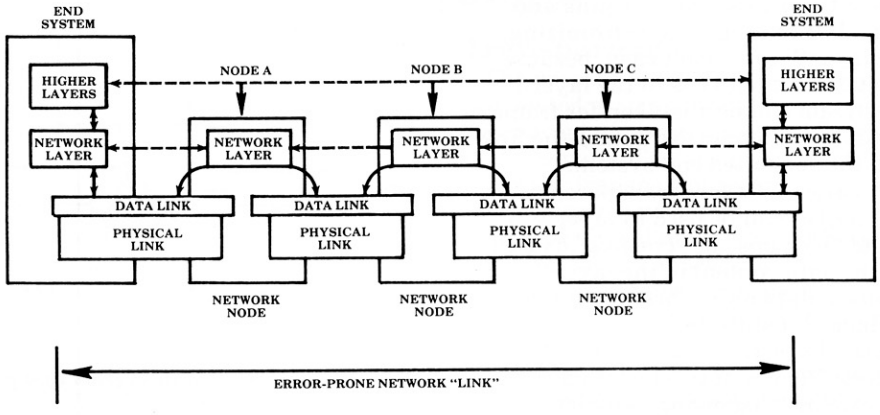


Figure 9. Network layer

by a single data link, more often, switched subnetworks are used because they offer greater connectivity and reliability at reduced cost. When a switched subnetwork is used, multiple data links must be "strung together," or "concatenated," to provide a single "network connection" between computers. In essence, the network layer takes the "perfect" data links provided by multiple data-link layers and strings these data links together so that they behave as a single "network link," or "connection" (Figure 9). The X.25 Packet-Level Protocol (PLP) is an example of a network-layer protocol that is used to provide "network connections" across public data networks (PDNs).

When end systems are not attached to a common subnetwork, their connection will require the use of more than one subnet. If so, the network layer may be divided into sublayers, with the lower sublayer managing "intra-network" connections as previously described and the upper sublayer managing "inter-network" connections (Figure 10). The "upper sublayer" works in much the same way as the network layer except that it handles the "concatenation" of multiple networks to create a single "inter-network link." Internet Protocol (IP) used in the ARPANET and the Defense Data Network (DDN) is an example of a protocol that manages "inter-network" connections.

Layer 4—the transport layer. The transport layer provides error control across the network connection provided by layer 3 regardless of whether the connection is "intra-network" or "inter-network" (Figure 11). Error control is needed across inter-network or intra-network connections because the concatenation of "perfect" data links

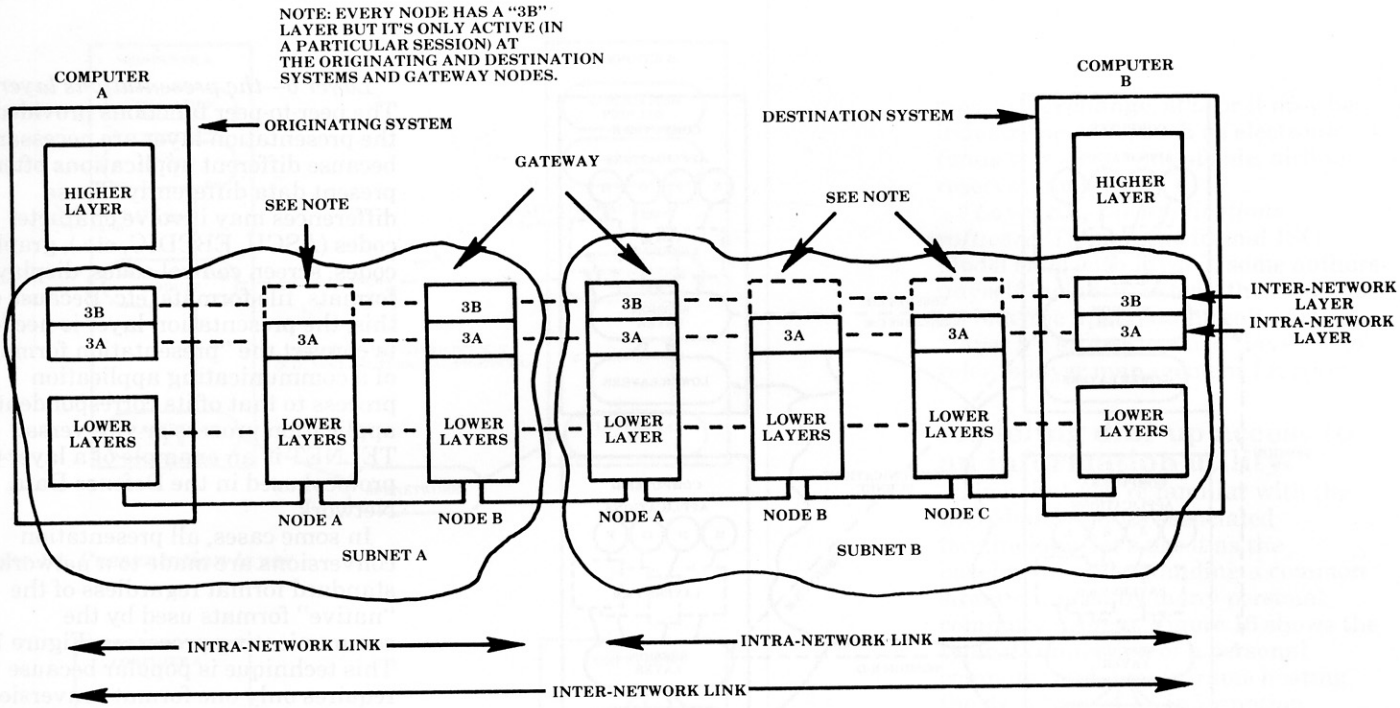


Figure 10. Upper sublayer

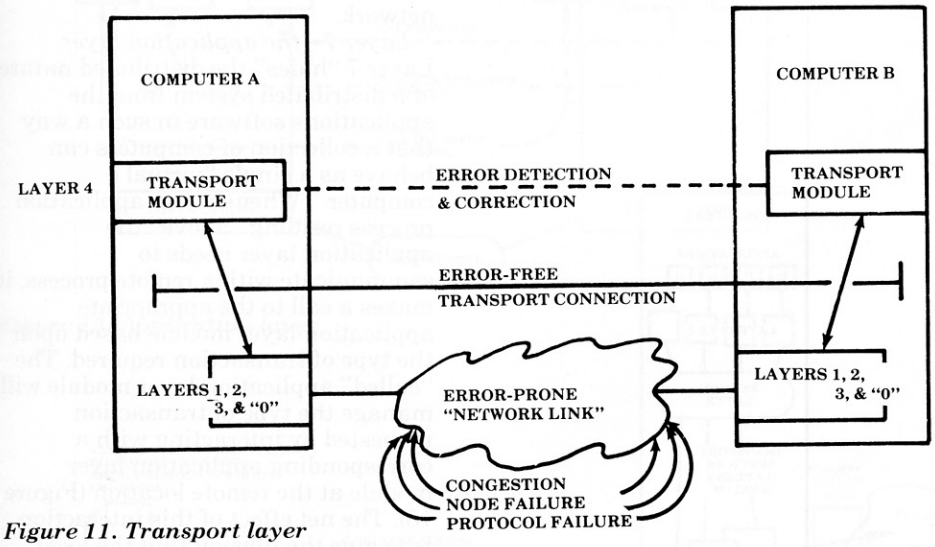


Figure 11. Transport layer

provided by layer 3 is, as a whole, subject to errors for a variety of reasons: subnetwork congestion, protocol failure, node failure, etc. Layer 4's job is to control these errors through some type of detection and correction. Conceptually, layer 4 operates "above" the network layer to convert an "error-prone network connection" into an "error-free transport connection."

Note that though the operation of layer 4 is analogous to that of layer 2, the scope of operation is different. Layer 2 "perfects" a single physical link; layer 4, however, "perfects" a "network connection" made up of many nodes connected by multiple physical links with their associated data links. Transmission Control Protocol (TCP) is an example of a layer-4 protocol that provides error control across network connections in the ARPANET and in the Defense Data Network (DDN).

Layer 5—the session layer. The peer-to-peer functions provided by the session layer are needed because of the existence of multiple, multi-user computers in a distributed computer network. In such a network, any computer may be host to many application processes, which in turn communicate across more than one transport connection with other processes running on different computers. Because of this, the session layer is needed to simultaneously manage multiple "sessions" between multiple application processes running on multiple computers (Figure 12).

From a vertical perspective, layer 5 "interfaces" a variety of user applications (layer 6 and above) to a variety of transport services (Figure 13). This is necessary because different applications require different "classes of service." For example, a file transfer requires a higher throughput than a database query. To accommodate such differences, the session layer is able to request different classes of service to support different types of data transfers.

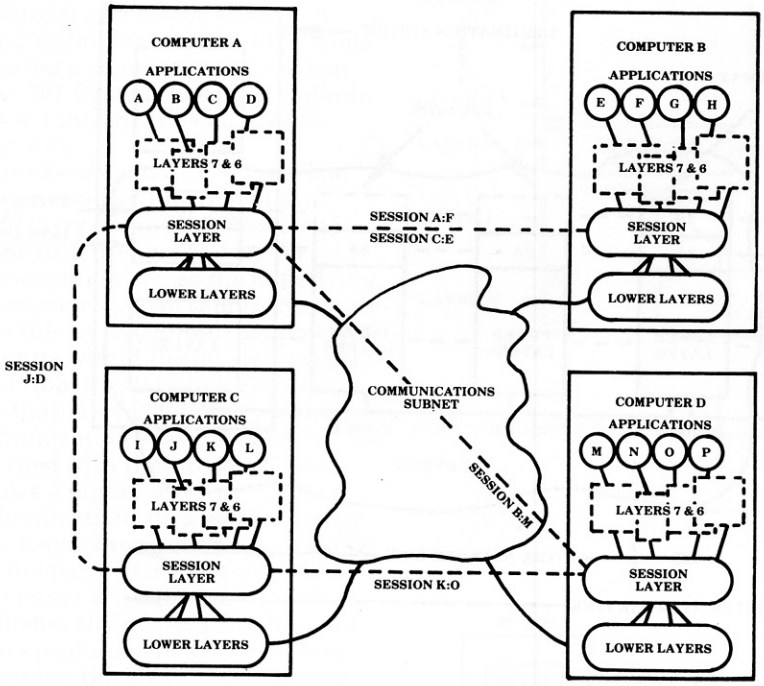


Figure 12. Session layer

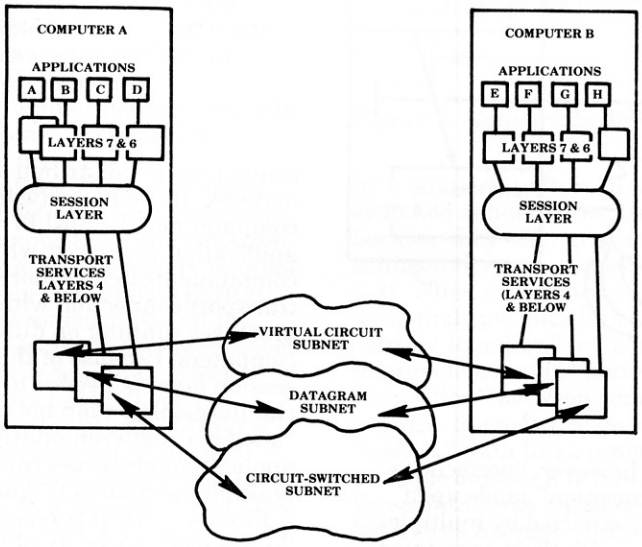


Figure 13. Interface to transport services

Layer 6—the presentations layer. The peer-to-peer functions provided by the presentation layer are necessary because different applications often present data differently. These differences may involve character codes (ASCII, EBCDIC, etc.), graphics codes, screen control codes, display formats, file formats, etc. Because of this, the presentation layer is needed to convert the "presentation format" of a communicating application process to that of its correspondent application process or vice versa. TELNET is an example of a layer-6 protocol used in the Defense Data Network.

In some cases, all presentation conversions are made to a network standard format regardless of the "native" formats used by the communicating processes (Figure 14). This technique is popular because it requires only one format conversion at each computer (from native to network standard) regardless of the number of different presentation formats in use throughout the network.

Layer 7—the application layer. Layer 7 "hides" the distributed nature of a distributed system from the applications software in such a way that a collection of computers can behave as a single "virtual computer." Whenever an application process residing "above" the application layer needs to communicate with a remote process, it makes a call to the appropriate application-layer module based upon the type of transaction required. The "called" application-layer module will manage the type of transaction requested by interacting with a corresponding application-layer module at the remote location (Figure 15). The net effect of this interaction is to give the illusion that the local and remote application processes reside on the same computer.

From the vertical perspective, the application layer gives applications software access to the OSI environment, or, in other words, access to a distributed computer environment. From the horizontal perspective, the peer-to-peer functions in the application layer may be provided by several application-layer modules because various applications may be involved with different types of "transactions." The type of transaction may be generic, such as a file transfer, document transfer,

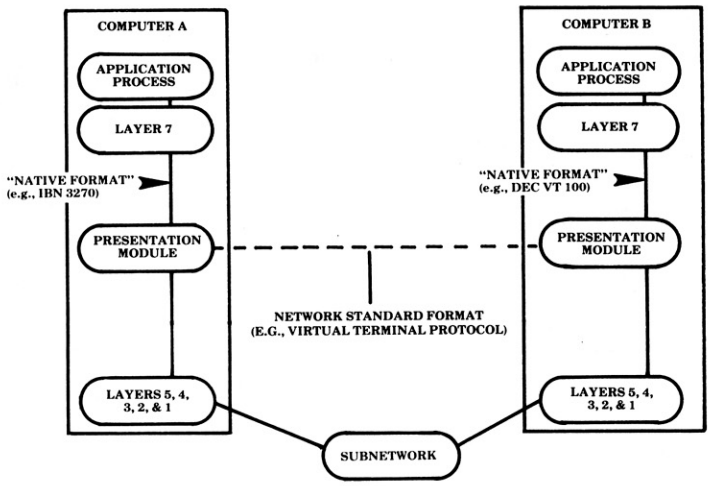


Figure 14. Presentation layer

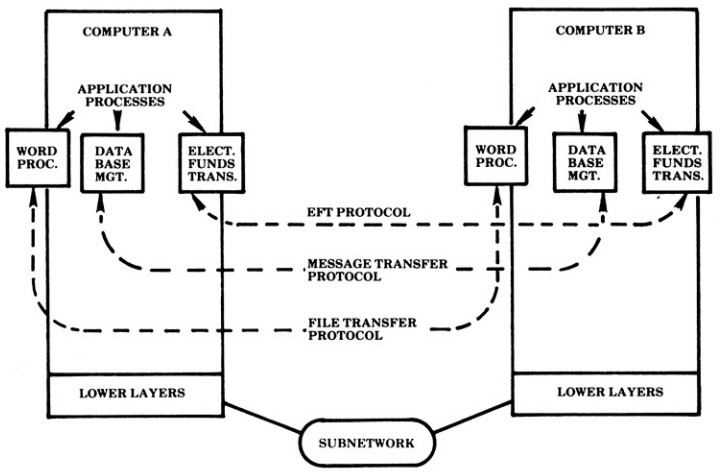


Figure 15. Application layer

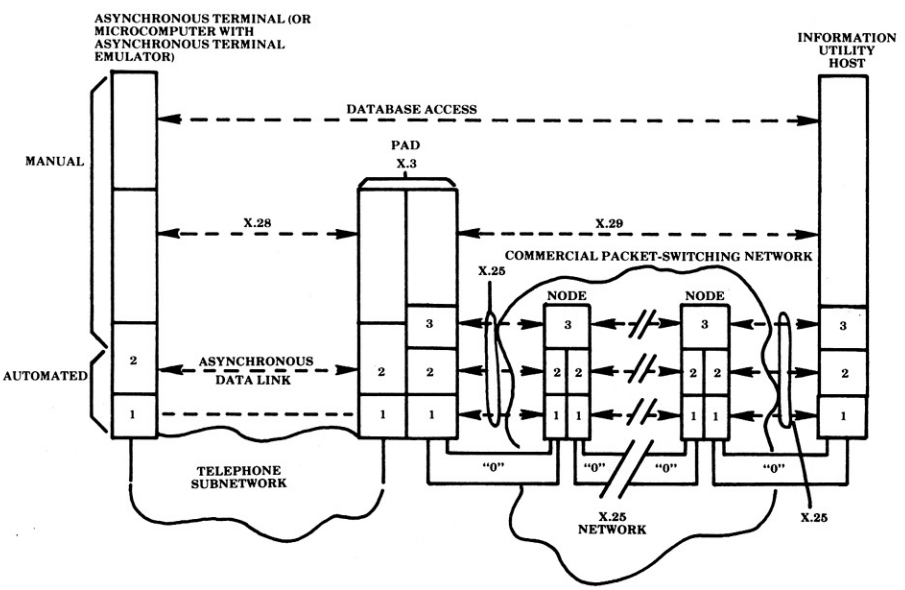


Figure 16. PC/mainframe connection

message exchange, etc.; or it may be industry-specific, such as electronic funds transfer, point-of-sale, airline reservations, etc.

"Layer 8"—the applications software. Though the formal ISO Model ends with layer 7, some authors (myself included) extend the model to include the applications software as "layer 8"; still others use "layer 8" to refer to layer management functions.

Modeling dial-up access to an information utility

Now that you're familiar with the ISO Model and its associated terminology, let's use it as the baseline in understanding a common situation faced by many personal computer owners. Figure 16 shows the typical connection of a personal computer and a mainframe hosting the data base of an information utility. Even though this connection isn't patterned after the ISO Model, we can still depict much of the functional architecture in terms of the model. Doing this helps us to communicate more effectively about the architecture and to understand it more completely.

The connections in Figure 16 involve multiple subnetworks, a "gateway," and two end systems. The subnetwork on the left is a circuit-switched network providing the physical connection between the personal computer and the PAD; the subnetwork on the right is a packet-switched network providing the connection between the PAD and the mainframe. The personal computer communicates with the PAD by emulating an asynchronous terminal. In this mode, the level of automation provided by the personal computer is limited, with some functions being provided manually by the personal computer operator. The PAD provides a "gateway" from the personal computer environment into that of the mainframe. As you can see, the PAD connects to a commercial packet-switched network at layers 3 and below, and connects with the mainframe at the higher layers.

The connections shown in Figure 16 depend upon numerous standards. RS-232C specifies the interface between the personal computer and its modem; CCITT Standard X.28 governs the connection of the personal computer as an asynchronous terminal to the PAD;

Alternatives to open systems interconnection

Open systems interconnection is the goal of the ISO Reference Model and can be achieved by conforming to standard protocols developed for each layer. By following the standards, manufacturers will be able to produce a system that is "open" to all other systems adhering to the same standards. Since the model only provides a framework for the development of standards, various standards committees are busy standardizing the details necessary for implementing the model. At present, standards for the lower layers are maturing, and standards for the upper layers are rapidly being developed.

Fully implementing a truly open system is difficult and expensive, but rapid progress is being made. Most computer system vendors have or are working on their own layered architectures, and many are anxious to offer ISO Model compatibility. But until open systems become widely available, most people will be more familiar with various alternatives.

One of these alternatives, terminal emulators, connect incompatible systems with each other by allowing one of the systems to emulate a device that is compatible with the other. "Smart terminal emulators," such as IBM 3270 emulators, are often used in business environments for this purpose, while many personal computer owners have emulator programs that cause their machines to emulate "dumb terminals." The popularity of such programs is due, in part, to their ability to provide a cheap and easy, although primitive, way to communicate between systems having incompatible system software. I use the word "primitive" because their use often produces a connection with limited use.

Gateways represent another strategy for connecting incompatible systems. A full discussion of gateways is beyond the scope of this article, but I do want to put the various types into perspective.

Gateways can be simple or complex depending on the amount of incompatibility they are designed to overcome. From our discussion of the ISO Model, you can see that incompatibility at any layer can prevent the interconnection of systems. The level at which incompatibilities exist is the key to classifying gateways. The two major categories are "layer-3 gateways" and "higher-layer" gateways. Layer-3 gateways, as the name implies, are designed to connect systems that are incompatible up through layer 3; the higher layers (4-7) in each system are compatible. Layer-3 gateways include a variety of devices: "bridges" used in local area networks (LANs), internet protocol (IP) gateways used in military and research networks, and "X.75 gateways" used between commercial packet-switching networks. On the other hand, a higher-layer gateway is designed to connect systems that are incompatible at the higher layers. "Higher layers" is not an exact term, but systems that are incompatible at layer 4 are usually incompatible at layers 5-7 also. Thus, the term usually means "layers 4-7."

The accompanying article addresses a special type of gateway that is not often recognized as such. However, a packet-assembler/disassembler (PAD) can be considered a gateway for asynchronous terminals into commercial packet-switching networks and associated host computers. The PAD gets its name because it "assembles" characters from asynchronous terminals into packets for transmission through a packet-switched network and also "disassembles" packets received over the packet-switched network into characters for delivery to asynchronous terminals. The PAD is essentially an "intelligent" statistical multiplexer that implements higher-layer functions of the ISO Model. Including this device in the accompanying article provides a means for tying together the concepts of gateways, terminal emulators, and standards, all within the context of the ISO Network Reference Model. It is also similar to the Terminal Access Controller (TAC) used in the DDN.

X.3 covers the functions the PAD performs; X.25 contains layer 1, 2, and 3 specifications for connections to the packet-switched network; X.29 specifies the higher-layer rules for interconnecting the host and the PAD. All of this is complicated, but I think you can appreciate how a knowledge of the ISO Model helps a person understand such a complex situation. Since the concepts behind the model make it such a powerful tool for understanding distributed systems, it is rapidly becoming the baseline for understanding distributed architectures in general.

The connections shown in Figure 16 also provide good examples of transparency. The first example is the connection between the personal computer and the PAD through the circuit-switched subnetwork. The circuit-switched subnet has nodes and links and protocols of its own, but the personal computer and the PAD see this entire subnet as a single link. All of the channel multiplexing, inter-office signalling, etc. are completely hidden from the devices attached to the subnet. Even though they're there, they appear not to be and are thus "transparent."

Another example of transparency involves the downloading of a file from the mainframe to the personal computer. If, for example, an error correcting protocol is used, it will operate "end-to-end" between the personal computer and the mainframe. Consequently, the circuit-switched subnet, the packet-switched subnet, and the PAD will all be transparent to the error correction process operating in what would be layer 4 of the ISO Model.

As mentioned earlier, data transparency depends on the type of data-link protocol in use. Two different types are used to connect the microcomputer and the mainframe computer in Figure 16. The first type is a simple asynchronous data link between the microcomputer and the PAD. The second is a "bit-oriented" protocol between the packet-switching nodes and between the packet-switching network and the mainframe computer.

Though layer 2 is important in achieving data transparency, data transparency can be affected by other layers also. Fixed-length characters are the layer-2 "protocol data units" for asynchronous data links and, as such, do not cause data-link disruptions due to the appearance of arbitrary bit patterns within the

character. But on asynchronous links, the transmission system can affect data transparency if it includes a modem that accepts commands embedded in the bit stream. This occurs because such modems are designed to recognize commands prefaced with some type of delimiter. Consequently, the data sent to such a modem can no longer contain an arbitrary bit pattern because such data might be mistaken for modem commands. When this happens, the modem will erroneously shift from the transmission to the command mode, thereby disrupting the link. To work around this problem, some communications programs provide filters that can be used to alter offending characters or symbols within a file. If this is done, the receiving communications program must have the ability to return the altered characters or symbols to their original condition.

Space doesn't permit further analysis of the network in Figure 16. But I think you can appreciate how a knowledge of the ISO Model allows you to focus in on the area under consideration without being distracted by extraneous information. With this introduction and a little extra study, you can sharpen your analytical skills tremendously. Let me conclude with a few examples of how useful the ISO Model can be as an analytic tool.

Analog versus digital communication. What effect will switching from analog to digital transmission equipment have on a system's architecture? According to the ISO Model, the effect of this change should be limited to the transmission system (layer "0") and the interface to it (layer 1). Layers 2 and above shouldn't be affected since the type of transmission system should be made "transparent" to higher layers by layer 1.

Another way of approaching this same question is to focus in on the only layer that deals with physical communications between peer entities, "layer 0." Since the peer entities in all other layers communicate by "virtual communications," "layer 0" and the interface to "layer 0" (layer 1) are obviously the areas affected by changes in the type of transmission equipment.

Synchronous versus asynchronous transmission. A similar question could be posed concerning the effect of switching from asynchronous to synchronous communication. This change would require different modems in the transmission system and changes in the interfaces to the modems. Layer 2 would also be affected because the layer 2 protocol data unit for asynchronous communication is a "character," whereas the protocol data unit for synchronous communication is a "block" or "frame." Layers 3 and above shouldn't be affected however. In fact, layer 3 in a well-designed system should be able to switch packets from asynchronous links to synchronous links without regard to whether the link is synchronous or asynchronous.

Types of encryption. The ISO Model is also extremely helpful in understanding the various levels of encryption in distributed systems and the respective advantages and disadvantages of each level. Typical levels include physical-level encryption employed in the transmission system (often erroneously referred to as data-link encryption), end-to-end encryption in layer 4, and user-to-user encryption in layer 6. By analyzing the effects of each level on the protocol data units described by the model, you'll be able to see that each encryption level presents a different degree of data security and key management complexity.

Physical channels, logical channels, and multiplexing. The ISO Model also helps us to understand the different types of logical channels and associated multiplexing. In terms of the model, multiplexing and logical channels are arranged in hierarchy. At the lowest level, the transmission equipment ("layer 0") typically multiplexes bits from many different "physical channels" over the same physical circuit. Higher up in the architecture, layer 2 can multiplex frames (representing logical channels at the data-link level) over the same physical channel. At the next level, layer 3 can multiplex packets from many different "network connections" (network-level logical connections) over the same data link. At an even higher level, layer 4 can multiplex messages from many different "transport connections" (transport-level logical channels) over the same network connection. Finally, layer 5 can multiplex sessions supporting many different end users (user-to-user logical channels) over the same transport connection. All of this is complex even if you have an understanding of the ISO Model; but without it, there's little hope of comprehending anything, even at a superficial level.

Clearly, the ISO Model is a valuable tool for understanding the complex distributed systems of the future. Every professional communicator needs to understand it because these systems will play a large role in our futures. Unfortunately, space doesn't permit me to give you all the information needed on the subject; but with a little extra study, you'll surprise yourself and others at how quickly you've learned to "talk ISO."

"ISO definitions" are on the following page.

ISO definitions

"Layer 0" converts bits into transmission symbols and propagates these symbols over a physical medium.

Layer 1 provides a physical connection across a link and interfaces the data terminal equipment (DTE) to the data communications equipment (DCE). The protocol data unit for the physical layer is the bit stream.

Layer 2 converts an "error-prone physical link" to an "error-free data link." The typical protocol data unit for the data-link layer is the "frame"; the main purpose of the frame is to transfer its contents (i.e., a packet) across a link without error.

Layer 3 (lower sublayer) concatenates "data links" to create a "intra-network link." The protocol data unit for the lower sublayer of the network layer is the "intra-network packet"; the main purpose of the intra-network packet is to route its contents (i.e., an "inter-network packet) across multiple data links from originating to destination nodes within a single subnetwork.

Layer 3 (upper sublayer) concatenates "intra-network links" to create an "inter-network link." The protocol data unit for the upper sublayer of the network layer is the "inter-network packet"; the main purpose of the inter-network packet is to route its contents (i.e., a message or portions of a message) across multiple "network links" from originating to destination nodes residing in different subnetworks.

Layer 4 converts an "error-prone intra- or inter-network link" to an "error-free end-to-end connection." The protocol data unit for the transport layer is the "message"; the main purpose of the message is to transfer the session-layer protocol data unit from one end system to another without error.

Layer 5 simultaneously manages multiple "dialogues" between multiple applications residing on multiple end systems. It interfaces applications to the appropriate "transport service." The main purpose of session-layer protocol data units is to provide satisfactory delivery of presentation-layer protocol data units to the correct presentation-layer entities.

Layer 6 manages the "form" of the data transferred between applications. The main purpose of presentation-layer protocol data units is to provide "presentable" application-layer protocol data units to application-entities.

Layer 7 manages the "type of data transfer." It interfaces applications software to a distributed computer environment. The main purpose of the application-layer protocol data unit is to provide an appropriate transfer of data between applications programs.

"Layer 8," the applications software, provides the "source" and "sink" for data transferred between end systems.

Advanced Data Communications Control Procedures (ADCCP) - See High-Level Data-Link Control.

American National Standards Institute (ANSI) - A voluntary U.S. standards organization involved in a wide variety of standardization activities, many involving computer networks.

asynchronous transmission - Transmission on a character-by-character basis in which the time intervals between characters can vary. Each character is synchronized by start and stop bits. Asynchronous transmission contrasts with synchronous transmission, in which the intervals between bits and characters are fixed through continuous synchronization.

architecture - The nature of an object as determined by its components, the attributes of the components, and the relationship between components.

Binary Synchronous Communication (BSC or BISYNC) - An IBM-defined, character-oriented, data-link protocol.

bit-oriented protocol - A protocol that implements its functions using bit sequences that are independent of any particular character code, such as ASCII or EBCDIC. Such codes are more efficient than character-oriented protocols that implement their control functions with characters. Also, bit-oriented protocols provide flexibility in system design, modification, and operation because they are independent of the "native" character code used in an end system. Character-oriented protocols do not provide the same degree of flexibility or efficiency.

bridge - A simple type of gateway. The simplest type of bridge connection is through address-space conversion. See gateway.

character-oriented protocols - See bit-oriented protocols.

circuit-switched network - A communications network that connects end systems by providing a dedicated physical connection across the network for the duration of the session between end systems.

classes of service - Unique sets of service parameters that include response time, security, integrity, throughput, cost, etc.

Comite Consultatif Internationale de Telegraphic Telephonie (CCITT) - A committee under the auspices of the International Telecommunications Union (ITU), a treaty-level organization under the United Nations. Several classes of members exist. Voting members are government representatives from member nations. The U.S. voting member is the Federal Communications Commission. CCITT has been active in the development of "X-series" standards for public data networks. In the past, CCITT and ISO standardization activities have conflicted, but CCITT recently agreed to follow the ISO Reference Model in future standardization efforts.

cyclic redundancy check (CRC) - An error detection process that allows the receiver of a data unit to detect errors by comparing the results of a mathematical process executed by the sender to its own results after executing the same mathematical process. A mismatch in results indicates an error occurred during transmission. The error is then corrected by retransmission.

data communications equipment (DCE) - Provides the interface between data terminal equipment (DTE) and a transmission system. The most common DCE is a modem. DTE ranges from simple devices implementing data-link functions to large-scale computers.

data terminal equipment (DTE) - See data communication equipment.

dumb terminal - See "smart terminal."

Electronic Industries Association (EIA) - A trade group engaged in standardizing electrical and functional characteristics of communications interface equipment. The most well-known standard produced by this group is RS-232C. (RS stands for "recommended standard.) This standard applies to the interface between a computer and a modem but has often been adapted for a variety of other purposes, such as the interface between a computer and a printer.

gateway - A facility or facilities for connecting dissimilar networks through some type of protocol conversion.

header - A common term for protocol control information. See "trailer."

High-Level Data-Link Control (HDLC) - A bit-oriented, data-link protocol defined by ISO. It is similar to ANSI's ADCCP and IBM's SDLC.

International Standards Organization (ISO) - A private international organization devoted to the development of a wide variety of standards from photographic film to data communications. Its primary membership includes representatives from national standards bodies. The U.S. representative is the American National Standards Institute (ANSI).

message-switched network - A communications network that provides communications between end systems by storing and forwarding messages from node to node across the network.

node - A point within a network where communication links converge. Nodes can be of two types. Intermediate nodes provide switching of data between endpoint nodes. The type of switching varies; for example, the node may switch circuits, bits, bytes, packets, or messages depending upon the particular type of network. Endpoint nodes provide the sources and sinks for data to and from the network.

packet-switched network - A communications network that provides communications between end systems by holding and forwarding packets of data from node to node across the network. A packet-switched network differs from a message-switched network in that the former network is not concerned with messages as a whole.

smart terminal - A terminal that has inherent processing power, in contrast to a "dumb terminal," which does not. A dumb terminal is essentially the equivalent of an asynchronous teletypewriter. Keyboard video display terminals (KVDT) in this category are sometimes referred to as "glass teletypes."

subnetwork - A network that is part of another network. Packet-switched networks, circuit-switched networks, and message-switched networks are often used to provide communications for computer networks. In this context, the switched networks are subnetworks within the overall computer network. Whether these switched networks are referred to as "networks" or "subnetworks" depends upon the immediate frame of reference.

Synchronous Data-Link Control (SDLC) - See High-Level Data-Link Control.

synchronous transmission - See asynchronous transmission.

topology - The physical arrangement of nodes, links, and end systems in a network. The topology of a network can be expanded into the network's "physical architecture" by identifying the actual hardware, firmware, and software specifications for each element of the network.

trailer - A common term for protocol control information located at the end of a protocol data unit. Typically, the data-link layer adds both a "header" and "trailer" to the "user data" from the network layer to allow it to "synchronize" the data unit with the receiving data-link module. Without this synchronization, the receiving data link wouldn't know where the data unit begins or ends.

transmission symbols - Symbols created through changes in the frequency, phrase, or amplitude of an electrical wave.

transport service - Refers to layers 1-4 collectively as opposed to "transport layer" which refers to layer 4 only. It is a useful term because the services provided by layer 4 to layer 5 are based on the capabilities of layers 1-4 rather than just layer 4 alone. Consequently, the selection of a particular class of service from layer 4 usually represents the implied selection of certain lower-layer services also.

X.75 - A CCITT standard that defines the layer 1-3 protocols for the gateways used to connect commercial packet-switched networks.

Capt. Brewer has 15 years experience in communications and is currently a systems training officer for Air Training Command at Randolph AFB, Texas. He has served as an instructor in computer networks for the Telecommunications Systems Staff Officer Course at Keesler AFB, Miss., and has taught graduate-level courses in computer communications for the University of Southern Mississippi. Before that, he was a requirements analyst for the Air Force Computer Communications Planning Center at Tinker AFB, Okla. He holds a master's degree in teleprocessing science and a bachelor's degree in industrial and vocational education.