OSI Reference Model Networking Layers, Sublayers and Layer Groupings

The most important OSI Reference Model concept is that of networking *layers*. It's not an exaggeration to say that layers are really the heart of the OSI model—the entire point of the model is to separate networking into distinct functions that operate at different levels. Each layer is responsible for performing a specific task or set of tasks, and dealing with the layers above and below it. The rest of this section will deal with many of the different nuances of this layer orientation.

OSI Reference Model Layers

The OSI Reference Model is comprised of seven conceptual layers, each assigned a "ranking" number from one to seven. The layer number represents the position of the layer in the model as a whole, and indicates how "close" the layer is to the actual hardware used to implement a network. The first and lowest layer is the *physical layer*, which is where low-level signaling and hardware are implemented. The seventh and highest layer is the *application layer*, which deals with high-level applications employed by users: both end users and the operating system software.

You can see that as we proceed from the first layer to the seventh, we move up the *layer stack* and in so doing, increase our level of *abstraction*. This means that the higher a layer is in the stack, the more it deals with logical concepts and software, and the less it deals with the hardware of a network and the "nuts and bolts" of making it work.

The first layer is the most concrete, as it deals with the actual hardware of networks, and the specific methods of sending bits from one device to another. It is the domain of hardware engineers and signaling experts. The second layer is a bit more abstract but still deals with signaling and hardware. As you proceed through the third, fourth and subsequent layers, the technologies at those layers become increasingly abstract. By the time you reach the seventh layer, you are no longer dealing with hardware or even operating system concepts very much; you are in the realm of the user and high-level programs that rely on lower levels to do the "heavy lifting" for them.

OSI Reference Model Layer Groupings

The OSI Reference Model does not formally assign any relationship between groups of adjacent layers. However, to help explain how the layers work, it is common to categorize them into two *layer groupings*:

- **Lower Layers (Layers 1, 2, 3 and 4):** The lower layers of the model—*physical*, *data link*, *network* and *transport*—are primarily concerned with the formatting, encoding and transmission of data over the network. They don't care that much about what the data is or what it is being used for, just about moving it around. They are implemented in both hardware and software, with the transition from hardware to software occurring as you proceed up from layer 1 to layer 4.

- **Upper Layers (Layers 5, 6 and 7):** The higher layers of the model—*session*, *presentation* and *application*—are the ones that are concerned primarily with interacting with the user, and implementing the applications that run over the network. The protocols that run at higher layers are less concerned with the low-level details of how data gets sent from one place to another; they rely on the lower layers to provide delivery of data. These layers are almost always implemented as software running on a computer or other hardware device.

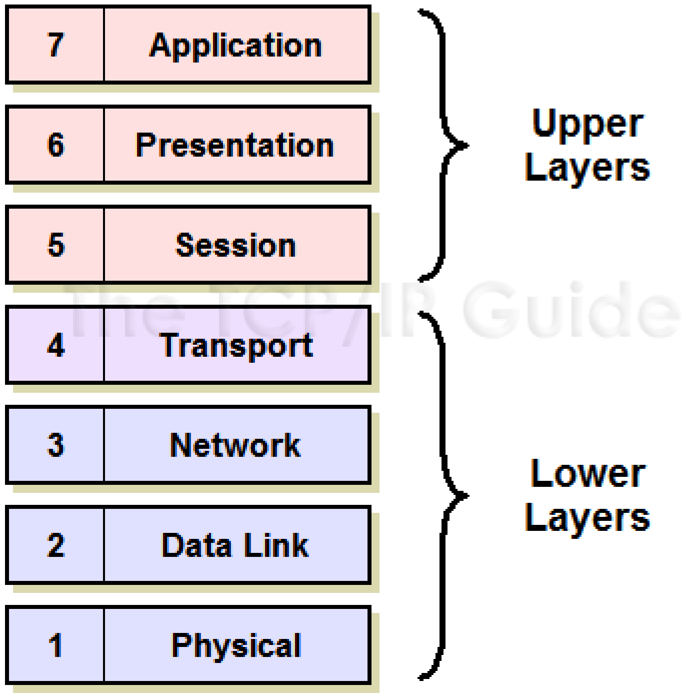

Figure 11: OSI Reference Model Layers

The OSI Reference Model divides networking functions into a stack of seven layers, numbered 1 through 7 from the bottom up. To help illustrate the differing levels of abstraction between layers near the top and those on the bottom, they are sometimes divided into two *layer groupings*—the *lower layers* and the *upper layers.* Of course, not everyone agrees on exactly how the division should be accomplished. In particular, the transport layer is sometimes considered an upper layer and sometimes a lower layer.

There are some who would not necessarily agree with how I have chosen to divide the layers above. In particular, valid arguments can be made for including the transport layer in the upper layer group, since it is usually implemented as software and is fairly abstract. I believe it is better as part of the lower layer group since its primary job is still providing services to higher layers for moving data, however. Really, layer 4 is somewhat of a "transition zone" and is hard to categorize. Figure 11 shows how I divide the OSI Reference Model layers into groups and indicates the special position of layer 4 in the stack.

Key Concept: The most fundamental concept in the OSI Reference Model is the division of networking functions into a set of *layers*, from layer one at the bottom to layer seven at the top. As you go up the layer stack, you move away from concrete, hardware-specific functions to ones that are increasingly abstract, until reaching the realm of user applications at layer seven. The seven layers are sometimes divided into groupings: the lower layers (one, two and three) and the upper layers (four through seven). There is some disagreement on whether layer four is a lower or upper layer.

Relationships Between OSI Reference Model Layers

There are also certain OSI layers that have "natural" relationships to each other. The physical and data link layers, in particular, are closely related. For example, most people talk about Ethernet as being a "layer two technology", but Ethernet specifications really deal with both layer 2 and layer 1. Similarly, layers three and four are often related; protocol suites are often designed so that layer three and four protocols work together; examples being TCP and IP in the TCP/IP protocol suite and IPX and SPX in the Novell suite.

In some areas, the layers are so closely related that the lines between them become *blurry*. This is particularly the case when looking at the higher layers; many technologies implement two or even all three of these layers, which is another reason why I feel they best belong in a group together. One important reason why the distinctions between layers five through seven are blurry is that the TCP/IP protocols are based on the TCP/IP model, which combines the functions of layers five through seven in a single, thick layer.

Key Concept: The four lower layers of the OSI model are most often discussed individually, because the boundaries between them are reasonably clear-cut. In contrast, the lines between the session, presentation and application layers are somewhat blurry. As a result, sometimes protocols span two or even all three of these layers; this is especially true of TCP/IP application protocols, since the TCP/IP model treats layers five through seven as a single layer.

Sublayers

Finally, note that some OSI Reference Model layers are further divided into *sublayers* to help define more precisely the internal details of protocols and technologies at those layers. This is commonly done at the lower layers, especially the physical layer and the data link layer.

OSI Reference Model Layers

Finally, after much ado—hopefully not ***too*** much!—it is time to take a look at the actual individual layers of the OSI Reference Model. As discussed in the section on OSI model concepts, each layer has certain characteristics that define it, and also various protocols normally associated with it. Understanding the nuances of each layer will help you understand all the technologies that use them.

In this section, I describe each of the OSI Reference Model layers individually. For each one I provide its name and layer number, describe its general function in the OSI layer stack, and outline the specific types of activities for which each is normally responsible. I also provide some examples of the technologies and protocols that reside at each layer. Keep in mind, however, that the descriptions in this section are *generic*. To really comprehend the various layers and how they are uses, there is no substitute for reading the details of the individual protocols that function at each layer, covered elsewhere in this Guide.

Related Information: For assistance in remembering the correct order of the layers, see the topic that describes common mnemonics used for the OSI Reference Model. To easily compare the key characteristics of the seven layers, refer to the OSI model layer summary.

Physical Layer (Layer 1)

The lowest layer of the OSI Reference Model is layer 1, the *physical layer*; it is commonly abbreviated "PHY". The physical layer is special compared to the other layers of the model, because it is the only one where data is physically moved across the network interface. All of the other layers perform useful functions to create messages to be sent, but they must all be transmitted down the protocol stack to the physical layer, where they are actually sent out over the network.

Note: The physical layer is also "special" in that it is the only layer that really does not apply specifically to TCP/IP. Even in studying TCP/IP, however, it is still important to understand its significance and role in relation to the other layers where TCP/IP protocols reside.

Understanding the Role of the Physical Layer

The name "physical layer" can be a bit problematic. Because of that name, and because of what I just said about the physical layer actually transmitting data, many people who study networking get the impression that the physical layer is only about actual network hardware. Some people may say the physical layer is "the network interface cards and cables". This is not actually the case, however. The physical layer defines a number of network functions, not just hardware cables and cards.

A related notion is that "all network hardware belongs to the physical layer". Again, this isn't strictly accurate. All hardware must have ***some*** relation to the physical layer in order to send data over the network, but hardware devices generally implement multiple layers of the OSI model, including the physical layer but also others. For example, an Ethernet network interface card performs functions at both the physical layer and the data link layer.

Physical Layer Functions

The following are the main responsibilities of the physical layer in the OSI Reference Model:

- **Definition of Hardware Specifications:** The details of operation of cables, connectors, wireless radio transceivers, network interface cards and other hardware devices are generally a function of the physical layer (although also partially the data link layer; see below).

- **Encoding and Signaling:** The physical layer is responsible for various encoding and signaling functions that transform the data from bits that reside within a computer or other device into signals that can be sent over the network.

- **Data Transmission and Reception:** After encoding the data appropriately, the physical layer actually transmits the data, and of course, receives it. Note that this applies equally to wired and wireless networks, even if there is no tangible cable in a wireless network!

- **Topology and Physical Network Design:** The physical layer is also considered the domain of many hardware-related network design issues, such as LAN and WAN topology.

In general, then, physical layer technologies are ones that are at the very lowest level and deal with the actual ones and zeroes that are sent over the network.

For example, when considering network interconnection devices, the simplest ones operate at the physical layer: repeaters, conventional hubs and transceivers. These devices have absolutely no knowledge of the contents of a message. They just take input bits and send them as output. Devices like switches and routers operate at higher layers and look at the data they receive as being more than voltage or light pulses that represent one or zero.

Relationship Between the Physical Layer and Data Link Layer

It's important to point out that while the physical layer of a network technology primarily defines the hardware it uses, the physical layer is closely related to the data link layer. Thus, it is not generally possible to define hardware at the physical layer "independently" of the technology being used at the data link layer. For example, Ethernet is a technology that describes specific types of cables and network hardware, but the physical layer of Ethernet can only be isolated from its data link layer aspects to a point. While Ethernet cables are "physical layer", for example, their maximum length is related closely to message format rules that exist at the data link layer.

Furthermore, some technologies perform functions at the physical layer that are normally more closely associated with the data link layer. For example, it is common to have the physical layer perform low-level (bit level) repackaging of data link layer frames for transmission. Error detection and correction may also be done at layer 1 in some cases. Most people would consider these "layer two functions".

In many technologies, a number of physical layers can be used with a data link layer. Again here, the classic example is Ethernet, where dozens of different physical layer implementations exist, each of which uses the same data link layer (possibly with slight variations.)

Physical Layer Sublayers

Finally, many technologies further subdivide the physical layer into *sublayers*. In order to increase performance, physical layer encoding and transmission methods have become more complex over time. The physical layer may be broken into layers to allow different network media to be supported by the same technology, while sharing other functions at the physical layer that are common between the various media. A good example of this is the physical layer architecture used for Fast Ethernet, Gigabit Ethernet and 10-Gigabit Ethernet.

Note: In some contexts, the physical layer technology used to convey bits across a network or communications line is called a *transport method*. Don't confuse this with the functions of the OSI transport layer (layer 4).

Key Concept: The lowest layer in the OSI Reference Model is the *physical layer*. It is the realm of networking hardware specifications, and is the place where technologies reside that perform data encoding, signaling, transmission and reception functions. The physical layer is closely related to the data link layer.

Data Link Layer (Layer 2)

The second-lowest layer (layer 2) in the OSI Reference Model stack is the *data link layer*, often abbreviated "DLL" (though that abbreviation has other meanings as well in the computer world). The data link layer, also sometimes just called the *link layer*, is where many wired and wireless local area networking (LAN) technologies primarily function. For example, Ethernet, Token Ring, FDDI and 802.11 ("wireless Ethernet" or "Wi-Fi') are all sometimes called "data link layer technologies". The set of devices connected at the data link layer is what is commonly considered a simple "network", as opposed to an internetwork.

Data Link Layer Sublayers: Logical Link Control (LLC) and Media Access Control (MAC)

The data link layer is often conceptually divided into two sublayers: *logical link control (LLC)* and *media access control (MAC)*. This split is based on the architecture used in the IEEE 802 Project, which is the IEEE working group responsible for creating the standards that define many networking technologies (including all of the ones I mentioned above except FDDI). By separating LLC and MAC functions, interoperability of different network technologies is made easier, as explained in our earlier discussion of networking model concepts.

Data Link Layer Functions

The following are the key tasks performed at the data link layer:

- **Logical Link Control (LLC):** Logical link control refers to the functions required for the establishment and control of logical links between local devices on a network. As mentioned above, this is usually considered a DLL sublayer; it provides services to the network layer above it and hides the rest of the details of the data link layer to allow different technologies to work seamlessly with the higher layers. Most local area networking technologies use the IEEE 802.2 LLC protocol.

- **Media Access Control (MAC):** This refers to the procedures used by devices to control access to the network medium. Since many networks use a shared medium (such as a single network cable, or a series of cables that are electrically connected into a single virtual medium) it is necessary to have rules for managing the medium to avoid conflicts. For example. Ethernet uses the CSMA/CD method of media access control, while Token Ring uses token passing.

- **Data Framing:** The data link layer is responsible for the final encapsulation of higher-level messages into *frames* that are sent over the network at the physical layer.
- **Addressing:** The data link layer is the lowest layer in the OSI model that is concerned with addressing: labeling information with a particular destination location. Each device on a network has a unique number, usually called a *hardware address* or *MAC address*, that is used by the data link layer protocol to ensure that data intended for a specific machine gets to it properly.
- **Error Detection and Handling:** The data link layer handles errors that occur at the lower levels of the network stack. For example, a cyclic redundancy check (CRC) field is often employed to allow the station receiving data to detect if it was received correctly.

Physical Layer Requirements Definition and Network Interconnection Device Layers

As I mentioned in the topic discussing the physical layer, that layer and the data link layer are very closely related. The requirements for the physical layer of a network are often part of the data link layer definition of a particular technology. Certain physical layer hardware and encoding aspects are specified by the DLL technology being used. The best example of this is the Ethernet standard, IEEE 802.3, which specifies not just how Ethernet works at the data link layer, but also its various physical layers.

Since the data link layer and physical layer are so closely related, many types of hardware are associated with the data link layer. Network interface cards (NICs) typically implement a specific data link layer technology, so they are often called “Ethernet cards”, “Token Ring cards”, and so on. There are also a number of network interconnection devices that are said to “operate at layer 2”, in whole or in part, because they make decisions about what to do with data they receive by looking at data link layer frames. These devices include most bridges, switches and barters, though the latter two also encompass functions performed by layer three.

Some of the most popular technologies and protocols generally associated with layer 2 are Ethernet, Token Ring, FDDI (plus CDDI), HomePNA, IEEE 802.11, ATM, and TCP/IP's Serial Link Interface Protocol (SLIP) and Point-To-Point Protocol (PPP).

Key Concept: The second OSI Reference Model layer is the *data link layer*. This is the place where most LAN and wireless LAN technologies are defined. Layer two is responsible for logical link control, media access control, hardware addressing, error detection and handling, and defining physical layer standards. It is often divided into the logical link control (LLC) and media access control (MAC) sublayers, based on the IEEE 802 Project that uses that architecture.

Network Layer (Layer 3)

The third-lowest layer of the OSI Reference Model is the *network layer*. If the data link layer is the one that basically defines the boundaries of what is considered a network, the network layer is the one that defines how *internetworks* (interconnected networks) function. The network layer is the lowest one in the OSI model that is concerned with actually getting data from one computer to another even if it is on a remote network; in contrast, the data link layer only deals with devices that are local to each other.

While all of layers 2 through 6 in the OSI Reference Model serve to act as "fences" between the layers below them and the layers above them, the network layer is particularly important in this regard. It is at this layer that the transition really begins from the more abstract functions of the higher layers—which don't concern themselves as much with data delivery—into the specific tasks required to get data to its destination. The transport layer, which is related to the network layer in a number of ways, continues this "abstraction transition" as you go up the OSI protocol stack.

Network Layer Functions

Some of the specific jobs normally performed by the network layer include:

- **Logical Addressing:** Every device that communicates over a network has associated with it a logical address, sometimes called a *layer three* address. For example, on the Internet, the Internet Protocol (IP) is the network layer protocol and every machine has an IP address. Note that addressing is done at the data link layer as well, but those addresses refer to local physical devices. In contrast, logical addresses are independent of particular hardware and must be unique across an entire internetwork.

- **Routing:** Moving data across a series of interconnected networks is probably the defining function of the network layer. It is the job of the devices and software routines that function at the network layer to handle incoming packets from various sources, determine their final destination, and then figure out where they need to be sent to get them where they are supposed to go. I discuss routing in the OSI model more completely in this topic on the topic on indirect device connection, and show how it works by way of an OSI model analogy.

- **Datagram Encapsulation:** The network layer normally encapsulates messages received from higher layers by placing them into *datagrams* (also called *packets*) with a network layer header.

- **Fragmentation and Reassembly:** The network layer must send messages down to the data link layer for transmission. Some data link layer technologies have limits on the length of any message that can be sent. If the packet that the network layer wants to send is too large, the network layer must split the packet up, send each piece to the data link layer, and then have pieces reassembled once they arrive at the network layer on the destination machine. A good example is how this is done by the Internet Protocol.

- **Error Handling and Diagnostics:** Special protocols are used at the network layer to allow devices that are logically connected, or that are trying to route traffic, to exchange information about the status of hosts on the network or the devices themselves.

Network Layer Connection-Oriented and Connectionless Services

Network layer protocols may offer either connection-oriented or connectionless services for delivering packets across the network. Connectionless ones are by far more common at the network layer. In many protocol suites, the network layer protocol is connectionless, and connection-oriented services are provided by the transport layer. For example, in TCP/IP, the Internet Protocol (IP) is connectionless, while the layer four Transmission Control Protocol (TCP) is connection-oriented.

The most common network layer protocol is of course the Internet Protocol (IP), which is why I have already mentioned it a couple of times. IP is the backbone of the Internet, and the foundation of the entire TCP/IP protocol suite. There are also several protocols directly related to IP that work with it at the network layer, such as IPsec, IP NAT and Mobile IP. ICMP is the main error-handling and control protocol that is used along with IP. Another notable network layer protocol outside the TCP/IP world is the Novell IPX protocol.

Key Concept: The OSI Reference Model's third layer is called the *network layer*. This is one of the most important layers in the model; it is responsible for the tasks that link together individual networks into *internetworks*. Network layer functions include internetwork-level addressing, routing, datagram encapsulation, fragmentation and reassembly, and certain types of error handling and diagnostics. The network layer and transport layer are closely related to each other.

The network interconnection devices that operate at the network layer are usually called *routers*, which at this point should hopefully come as no surprise to you. They are responsible for the routing functions I have mentioned, by taking packets received as they are sent along each “hop” of a route and sending them on the next leg of their trip. They communicate with each other using routing protocols, to determine the best routes for sending traffic efficiently. So-called “brouters” also reside at least in part at the network layer, as do the rather obviously named “layer three switches”. ☺

Transport Layer (Layer 4)

The fourth and “middle” layer of the OSI Reference Model protocol stack is the *transport layer*. I consider the transport layer in some ways to be part of both the lower and upper “groups” of layers in the OSI model. It is more often associated with the lower layers, because it concerns itself with the ***transport*** of data, but its functions are also somewhat high-level, resulting in the layer having a fair bit in common with layers 5 through 7 as well.

Recall that layers 1, 2 and 3 are concerned with the actual packaging, addressing, routing and delivery of data; the physical layer handles the bits; the data link layer deals with local networks and the network layer handles routing between networks. The transport layer, in contrast, is sufficiently conceptual that it no longer concerns itself with these “nuts and bolts” matters. It relies on the lower layers to handle the process of moving data between devices.

The transport layer really acts as a “liaison” of sorts between the abstract world of applications at the higher layers, and the concrete functions of layers one to three. Due to this role, the transport layer’s overall job is to provide the necessary functions to enable communication between software application processes on different computers. This encompasses a number of different but related duties

Modern computers are multitasking, and at any given time may have many different software applications all trying to send and receive data. The transport layer is charged with providing a means by which these applications can all send and receive data using the same lower-layer protocol implementation. Thus, the transport layer is sometimes said to be responsible for *end-to-end* or *host-to-host* transport (in fact, the equivalent layer in the TCP/IP model is called the “host-to-host transport layer”).

Transport Layer Services and Transmission Quality

Accomplishing this communication between processes requires that the transport layer perform several different, but related jobs. For transmission, the transport layer protocol must keep track of what data comes from each application, then combine this data into a single flow of data to send to the lower layers. The device receiving information must reverse these operations, splitting data and funneling it to the appropriate recipient processes. The transport layer is also responsible for defining the means by which potentially large amounts of application data are divided into smaller blocks for transmission.

Another key function of the transport layer is to provide *connection services* for the protocols and applications that run at the levels above it. These can be categorized as either connection-oriented services or connectionless services. Neither is better or worse than the other; they each have their uses. While connection-oriented services can be handled at the network layer as well, they are more often seen in the transport layer in the "real world". Some protocol suites, such as TCP/IP, provide both a connection-oriented and a connectionless transport layer protocol, to suit the needs of different applications.

The transport layer is also the place in the layer stack where functions are normally included to add features to end-to-end data transport. Where network layer protocols are normally concerned with just "best effort" communications, where delivery is not guaranteed. Transport layer protocols are given intelligence in the form of algorithms that ensure that reliable and efficient communication between devices takes place. This encompasses several related jobs, including lost transmission detection and handling, and managing the rate at which data is sent to ensure that the receiving device is not overwhelmed.

Transmission quality, meaning ensuring that transmissions are received as sent, is so important that some networking references define the transport layer on the basis of reliability and flow-control functions. However, not all transport layer protocols provide these services. Just as a protocol suite may have a connection-oriented and a connectionless transport layer protocol, it may also have one that provides reliability and data management services, and one that does not. Again, this is the case with TCP/IP: there is one main transport layer protocol, TCP, that includes reliability and flow control features, and a second, UDP, that doesn't.

Transport Layer Functions

Let's look at the specific functions often performed at the transport layer in more detail:

- **Process-Level Addressing:** Addressing at layer two deals with hardware devices on a local network, and layer three addressing identifies devices on a logical internetwork. Addressing is also performed at the transport layer, where it is used to differentiate between software programs. This is part of what enables many different software programs to use a network layer protocol simultaneously, as mentioned above. The best example of transport-layer process-level addressing is the TCP and UDP port mechanism used in TCP/IP, which allows applications to be individually referenced on any TCP/IP device.

- **Multiplexing and Demultiplexing:** Using the addresses I just mentioned, transport layer protocols on a sending device *multiplex* the data received from many application programs for transport, combining them into a single stream of data to be sent. The same protocols receive data and then *demultiplex* it from the incoming stream of datagrams, and direct each package of data to the appropriate recipient application processes.

- **Segmentation, Packaging and Reassembly:** The transport layer segments the large amounts of data it sends over the network into smaller pieces on the source machine, and then reassemble them on the destination machine. This function is similar conceptually to the fragmentation function of the network layer; just as the network layer fragments messages to fit the limits of the data link layer, the transport layer segments messages to suit the requirements of the underlying network layer.

- **Connection Establishment, Management and Termination:** Transport layer connection-oriented protocols are responsible for the series of communications required to establish a connection, maintain it as data is sent over it, and then terminate the connection when it is no longer required.

- **Acknowledgments and Retransmissions:** As mentioned above, the transport layer is where many protocols are implemented that guarantee reliable delivery of data. This is done using a variety of techniques, most commonly the combination of *acknowledgments* and *retransmission timers*. Each time data is sent a timer is started; if it is received, the recipient sends back an acknowledgment to the transmitter to indicate successful transmission. If no acknowledgment comes back before the timer expires, the data is retransmitted. Other algorithms and techniques are usually required to support this basic process.

- **Flow Control:** Transport layer protocols that offer reliable delivery also often implement *flow control* features. These features allow one device in a communication to specify to another that it must "throttle back" the rate at which it is sending data, to avoid bogging down the receiver with data. These allow mismatches in speed between sender and receiver to be detected and dealt with.

Relationship Between the Transport Layer and Network Layer

In theory, the transport layer and network layer are distinct, but in practice, they are often very closely related to each other. You can see this easily just by looking at the names of common protocol stacks—they are often named after the layer three and four protocols in the suite, implying their close relationship. For example, the name "TCP/IP" comes from the suite's most commonly used transport layer protocol (TCP) and network layer protocol (IP). Similarly, the Novell NetWare suite is often called "IPX/SPX" for its layer three (IPX) and layer four (SPX) protocols. Typically, specific transport layer protocols use the network layers in the same family. You won't often find a network using the transport layer protocol from one suite and the network layer protocol from another.

The most commonly used transport layer protocols are the Transmission Control Protocol (TCP) and User Datagram Protocol (UDP) in the TCP/IP suite, the Sequenced Packet Exchange (SPX) protocol in the NetWare protocol suite, and the NetBEUI protocol in the NetBIOS/NetBEUI/NBF suite (though NetBEUI is more difficult to categorize.)

Key Concept: The fourth and middle OSI Reference Model layer is the *transport layer*. This is another very important conceptual layer in the model; it represents the transition point between the lower layers that deal with data delivery issues, and the higher layers that work with application software. The transport layer is responsible for enabling *end-to-end communication* between application processes, which it accomplishes in part through the use of process-level addressing and multiplexing/demultiplexing. Transport layer protocols are responsible for dividing application data into blocks for transmission, and may be either connection-oriented or connectionless. Protocols at this layer also often provide data delivery management services such as reliability and flow control.

Session Layer (Layer 5)

The fifth layer in the OSI Reference Model is the *session layer*. As we proceed up the OSI layer stack from the bottom, the session layer is the first one where pretty much all practical matters related to the addressing, packaging and delivery of data are left behind—they are functions of layers four and below. It is the lowest of the three upper layers, which collectively are concerned mainly with software application issues and not with the details of network and internet implementation.

The name of this layer tells you much about what it is designed to do: to allow devices to establish and manage *sessions*. In general terms, a session is a persistent logical linking of two software application processes, to allow them to exchange data over a prolonged period of time. In some discussions, these sessions are called *dialogs*; they are roughly analogous to a telephone call made between two people.

Application Program Interfaces (APIs)

The primary job of session layer protocols is to provide the means necessary to set up, manage, and end sessions. In fact, in some ways, session layer software products are more sets of tools than specific protocols. These session-layer tools are normally provided to higher layer protocols through command sets often called *application program interfaces* or *APIs*.

Common APIs include NetBIOS, TCP/IP Sockets and Remote Procedure Calls (RPCs). They allow an application to accomplish certain high-level communications over the network easily, by using a standardized set of services. Most of these session-layer tools are of primary interest to the developers of application software. The programmers use the APIs to write software that is able to communicate using TCP/IP without having to know the implementation details of how TCP/IP works.

For example, the Sockets interface lies conceptually at layer five and is used by TCP/IP application programmers to create sessions between software programs over the Internet on the UNIX operating system. Windows Sockets similarly lets programmers create Windows software that is Internet-capable and able to interact easily with other software that uses that interface. (Strictly speaking, Sockets is not a protocol, but rather a programming method.)

Session Layer Functions

As I have mentioned in a few places in this Guide, the boundaries between layers start to get very fuzzy once you get to the session layer, which makes it hard to categorize what exactly belongs at layer 5. Some technologies really span layers 5 through 7, and especially in the world of TCP/IP, it is not common to identify protocols that are specific to the OSI session layer.

The term “session” is somewhat vague, and this means that there is sometimes disagreement on the specific functions that belong at the session layer, or even whether certain protocols belong at the session layer or not. To add to this potential confusion, there is the matter of differentiating between a “connection” and a “session”. Connections are normally the province of layer four and layer three, yet a Transmission Control Protocol (TCP) connection, for example, can persist for a long time. The longevity of TCP connections makes them hard to distinguish from “sessions” (and in fact there are some people who feel that the TCP/IP host-to-host transport layer really straddles OSI layers four and five).

Key Concept: The fifth layer in the OSI Reference Model layer is the *session layer*. As its name suggests, it is the layer intended to provide functions for establishing and managing sessions between software processes. Session layer technologies are often implemented as sets of software tools called *application program interfaces (APIs)*, which provide a consistent set of services that allow programmers to develop networking applications without needing to worry about lower-level details of transport, addressing and delivery.

Presentation Layer (Layer 6)

The *presentation layer* is the sixth layer of the OSI Reference Model protocol stack, and second from the top. It is different from the other layers in two key respects. First, it has a much more limited and specific function than the other layers; it's actually somewhat easy to describe, hurray! Second, it is used much less often than the other layers; in many types of connections it is not required.

The name of this layer suggests its main function as well: it deals with the *presentation* of data. More specifically, the presentation layer is charged with taking care of any issues that might arise where data sent from one system needs to be viewed in a different way by the other system. It also takes care of any special processing that must be done to data from the time an application tries to send it until the time it is sent over the network.

Presentation Layer Functions

Here are some of the specific types of data handling issues that the presentation layer handles:

- **Translation:** Networks can connect very different types of computers together: PCs, Macintoshes, UNIX systems, AS/400 servers and mainframes can all exist on the same network. These systems have many distinct characteristics and represent data in different ways; they may use different character sets for example. The presentation layer handles the job of hiding these differences between machines.

- **Compression:** Compression (and decompression) may be done at the presentation layer to improve the throughput of data. (There are some who believe this is not, strictly speaking, a function of the presentation layer.)

- **Encryption:** Some types of encryption (and decryption) are performed at the presentation layer. This ensures the security of the data as it travels down the protocol stack. For example, one of the most popular encryption schemes that is usually associated with the presentation layer is the Secure Sockets Layer (SSL) protocol. Not all encryption is done at layer 6, however; some encryption is often done at lower layers in the protocol stack, in technologies such as IPSec.

Presentation Layer Role in the OSI Model

The reason that the presentation layer is not always used in network communications is that the jobs mentioned above are simply not always needed. Compression and encryption are usually considered “optional”, and translation features are also only needed in certain circumstances. Another reason why the presentation layer is sometimes not mentioned is that its functions may be performed as part of the application layer.

The fact that the translation job done by the presentation layer isn't always needed means that it is common for it to be “skipped” by actual protocol stack implementations. This means that protocols at layer seven may talk directly with those at layer five. Once again, this is part of the reason why all of the functions of layers five through seven may be included together in the same software package, as described in the overview of layers and layer groupings.

Key Concept: The sixth OSI model layer is called the *presentation layer*. Protocols at this layer take care of manipulation tasks that transform data from one representation to another, such as translation, compression and encryption. In many cases, no such functions are required in a particular networking stack; if so, there may not be any protocol active at layer six.

Application Layer (Layer 7)

At the very top of the OSI Reference Model stack of layers, we find layer 7, the *application layer*. Continuing the trend that we saw in layers 5 and 6, this one too is named very appropriately: the application layer is the one that is used by network applications. These programs are what actually implement the functions performed by users to accomplish various tasks over the network.

It's important to understand that what the OSI model calls an “application” is not exactly the same as what we normally think of as an “application”. In the OSI model, the application layer provides services for user applications to employ. For example, when you use your Web browser, that actual software is an application running on your PC. It doesn't really “reside” at the application layer. Rather, it makes use of the services offered by a protocol that operates at the application layer, which is called the Hypertext Transfer Protocol (HTTP). The distinction between the browser and HTTP is subtle, but important.

The reason for pointing this out is because not all user applications use the application layer of the network in the same way. Sure, your Web browser does, and so does your e-mail client and your Usenet news reader. But if you use a text editor to open a file on another machine on your network, that editor is not using the application layer. In fact, it has no clue that the file you are using is on the network: it just sees a file addressed with a name that has been mapped to a network somewhere else. The operating system takes care of *redirecting* what the editor does, over the network.

Similarly, not all uses of the application layer are by applications. The operating system itself can (and does) use services directly at the application layer.

That caveat aside, under normal circumstances, whenever you interact with a program on your computer that is designed specifically for use on a network, you are dealing directly with the application layer. For example, sending an e-mail, firing up a Web browser, or using an IRC chat program—all of these involve protocols that reside at the application layer.

There are dozens of different application layer protocols that enable various functions at this layer. Some of the most popular ones include HTTP, FTP, SMTP, DHCP, NFS, Telnet, SNMP, POP3, NNTP and IRC. Lots of alphabet soup, sorry. ☺ I describe all of these and more in the chapter on higher-layer protocols and applications.

As the "top of the stack" layer, the application layer is the only one that does not provide any services to the layer above it in the stack—there isn't one! Instead, it provides services to programs that want to use the network, and to you, the user. So the responsibilities at this layer are simply to implement the functions that are needed by users of the network. And, of course, to issue the appropriate commands to make use of the services provided by the lower layers.

Key Concept: The seventh and highest layer in the OSI Reference Model is the *application layer*. Application protocols are defined at this layer, which implement specific user applications and other high-level functions. Since they are at the top of the stack, application protocols are the only ones that do not provide services to a higher layer; they make use of services provided by the layers below.

As we've discussed elsewhere, the distinctions between the top layers are not very clear, and this is largely because of the decision made to not separate out session, presentation and application layer functions in the important TCP/IP protocol suite. All of the protocols mentioned above are from the TCP/IP protocol family, and some may cover all three of the top three OSI layers, two of them, or one; in the TCP/IP model, they are all applications.

Data Encapsulation, Protocol Data Units (PDUs) and Service Data Units (SDUs)

Protocols are what describe the rules that control horizontal communication, that is, conversations between processes that run at corresponding layers within the OSI Reference Model. At every layer (except layer one) these communications ultimately take the form of some sort of message that is sent between corresponding software elements on two or more devices. Since these messages are the mechanism for communicating information between protocols, they are most generally called *protocol data units (PDUs)*. Each PDU has a specific format that implements the features and requirements of the protocol.

Layer Services and Data Encapsulation

As we've already discussed in our look at protocols, the communication between layers higher than layer one is *logical*; the only hardware connection is at the physical layer. Thus, in order for a protocol to communicate, it must pass down its PDU to the next lower layer for transmission. We've also already seen that using OSI terminology, lower layers are said to provide *services* to the layers immediately above them. One of the services each layer provides is this function: to handle and manage data received from the layer above.

At any particular layer N, a PDU is a complete message that implements the protocol at that layer. However, when this "layer N PDU" is passed down to layer N-1, it becomes the ***data*** that the layer N-1 protocol is supposed to ***service***. Thus, the layer N protocol data unit (PDU) is called the layer N-1 *service data unit (SDU)*. The job of layer N-1 is to transport this SDU, which it does in turn by placing the layer N SDU into its own PDU format, preceding the SDU with its own headers and appending footers as necessary. This process is called *data encapsulation*, because the entire contents of the higher-layer message are encapsulated as the data payload of the message at the lower layer.

What does layer N-1 do with its PDU? It of course passes it down to the next lower layer, where it is treated as a layer N-2 SDU. Layer N-2 creates a layer N-2 PDU containing the layer N-1 SDU and layer N-2's headers and footers. And the so the process continues, all the way down to the physical layer. In the theoretical model, what you end up with is a message at layer 1 that consists of application-layer data that is encapsulated with headers and/or footers from each of layers 7 through 2 in turn, as shown in Figure 15.

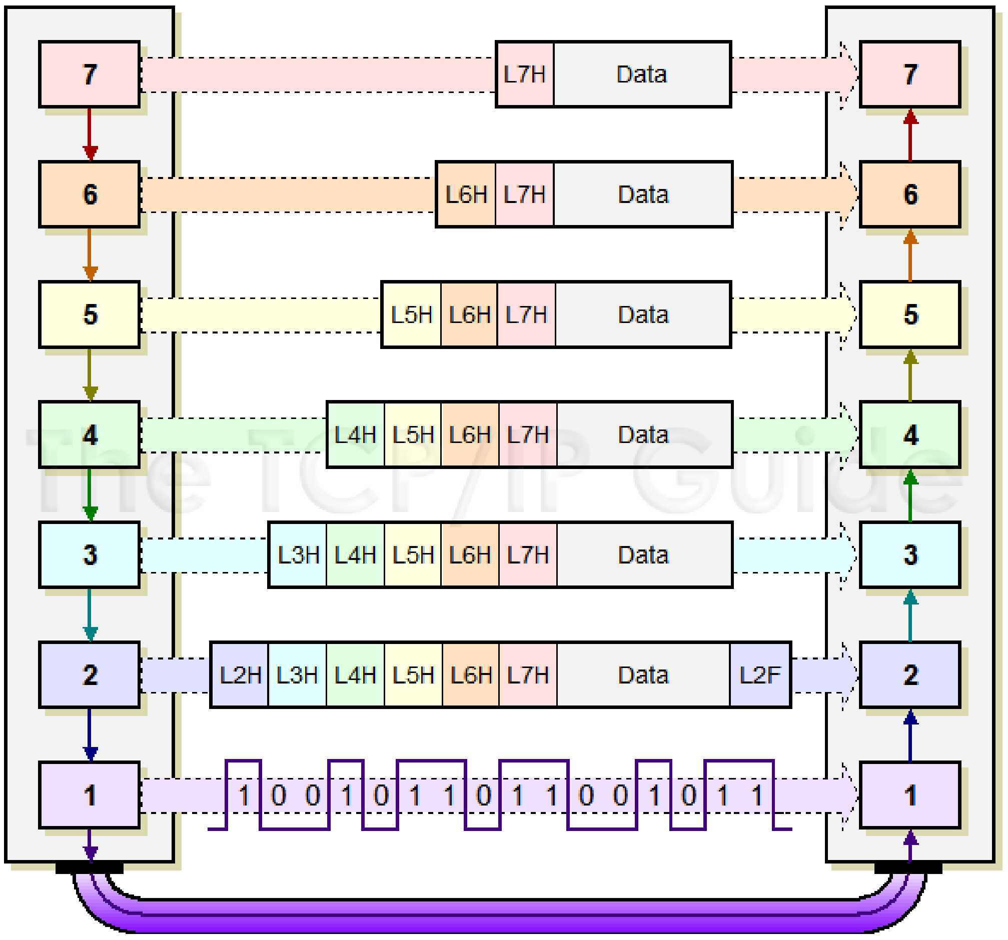

Figure 15: OSI Reference Model Data Encapsulation

Each protocol creates a *protocol data unit (PDU)* for transmission that includes headers required by that protocol and data to be transmitted. This data becomes the *service data unit (SDU)* of the next layer below it. This diagram shows a layer 7 PDU consisting of a layer 7 header ("L7H") and application data. When this is passed to layer 6, it becomes a layer 6 SDU. The layer 6 protocol prepends to it a layer 6 header ("L6H") to create a layer 6 PDU, which is passed to layer 5. The encapsulation process continues all the way down to layer 2, which creates a layer 2 PDU—in this case shown with both a header and a footer—that is converted to bits and sent at layer 1.

Data Encapsulation in TCP/IP

The "N-1, N-2" stuff makes this seem more difficult than it really is, so let's use a real-world (simplified) example instead. The Transmission Control Protocol (TCP) operates at layer 4 of the OSI model. It transmits messages called *segments* that contain data encapsulated from higher-layer protocols. The layer below TCP is the Internet Protocol (IP) at layer 3. It receives data from TCP and encapsulates it for transmission.

So, in the formal language of the OSI Reference Model, TCP segments are created as layer 4 PDUs. When passed to IP, they are treated as layer 3 SDUs. The IP software packages these SDUs into messages called *IP packets* or *IP datagrams*, which are layer 3 PDUs. These are in turn passed down to a layer 2 protocol, say Ethernet, which treats IP datagrams as layer 2 SDUs, and packages them into layer 2 PDUs (Ethernet frames) which are sent on layer 1. (Actually, in some technologies further encapsulation even occurs at layer one prior to transmission.)

On the receiving device, the process of encapsulation is reversed. The Ethernet software inspects the layer 2 PDU (Ethernet frame) and removes from it the layer 2 SDU (IP datagram) which it passes up to IP as a layer 3 PDU. The IP layer removes the layer 3 SDU (TCP segment) and passes it to TCP as a layer 4 PDU. TCP in turn continues the process, going back up the protocol layer stack. The complete process is illustrated in Figure 16.

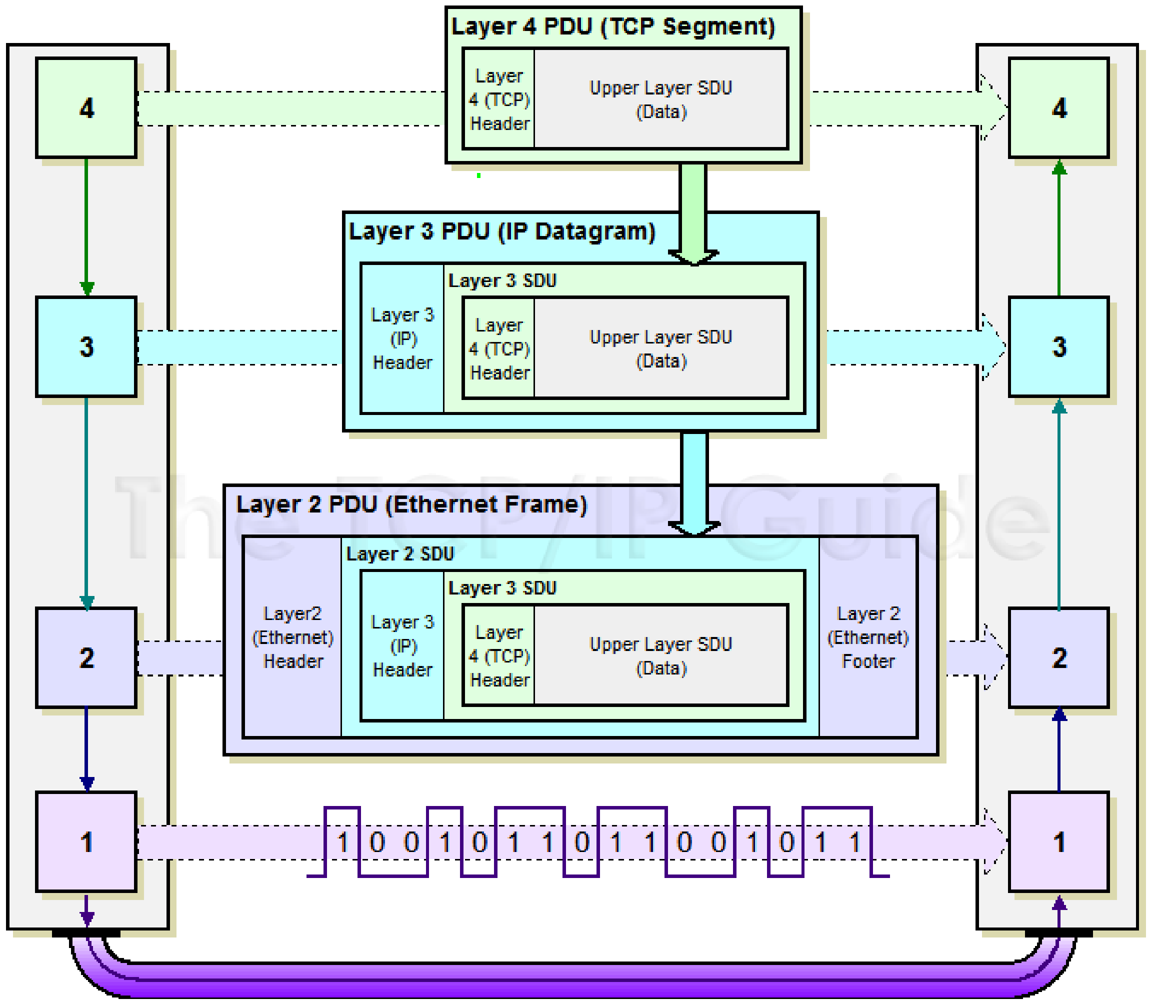

Figure 16: OSI Reference Model PDU and SDU Encapsulation

This example shows in more detail how OSI PDUs and SDUs are created and encapsulated. A TCP segment (layer 4 PDU) becomes a layer 3 SDU, which is encapsulated into a layer 3 PDU through the addition of an IP header. This becomes the payload of an Ethernet frame, which is a layer 2 PDU containing an Ethernet header, layer 2 SDU (the IP datagram) and Ethernet footer. The receiving device extracts the IP datagram from the Ethernet header and passes it to layer 3; the IP software extracts the TCP segment and passes it up to the TCP software.

This whole matter of passing data up and down the protocol stack, encapsulation and so on may seem needlessly complex. It also may appear to be rather inefficient; why send a message with so many headers and footer? However, the notion of data encapsulation is critical to creating modular, flexible networks.

Use of PDU and SDU Terminology

The term “protocol data unit” is rather formal. You will see it used in standards and sometimes in discussions, but more often, the “plain” message terms, such as “frame” and “datagram”, are encountered, as discussed in the networking fundamentals topic on messages. Similarly, data encapsulated by these messages is not normally called a “service data unit” but rather simply the *message body* or *payload*, as discussed in the topic on message formatting. There are cases, however, where knowing the difference between an SDU and a PDU is important to understanding the technology. One example is the IEEE 802.11 physical layer; the 802.11 standards talk about SDUs and PDUs constantly.

Related Information: See the OSI Reference Model analogy if you want to see an example that compares networking encapsulation to a type done in a real-world, non-networking context.

Key Concept: The message used to communicate information for a particular protocol is called its *protocol data unit (PDU)* in OSI model terminology. That PDU is passed down to the next lower layer for transmission; since that layer is providing the service of handling that PDU, it is called the lower layer’s *service data unit (SDU)*. The SDU is *encapsulated* into that layer’s own PDU and in turn sent to the next lower layer in the stack, proceeding until the physical layer is reached. The process is reversed on the recipient device. In summary: a layer N PDU is a layer N-1 SDU, which is encapsulated into a layer N-1 PDU.

Indirect Device Connection and Message Routing

Most of the explanations that I have provided in the other topics of this section have discussed the mechanisms by which machines connect to each other over a network *directly*. However, one of the most powerful aspects of networking is that it is possible to create internetworks—networks of networks—which allow devices to be connected *indirectly*. For example, machine “A” may send a message to machine “B” without really even knowing where it is on the network at all.

If a message is being sent between devices that are not on the same network, then it must be passed between directly-connected networks until it reaches its final destination. The process of transmitting a message from one network to another is called *forwarding*, and the collective process of forwarding from one device to another is *routing*. These concepts are fundamental to all internetworking, including the Internet itself. Every time you access an Internet resource such as a Web site, you are sending messages that get routed to that site, and the responses you receive get routed back.

Note: Even though the technically-correct term for moving a message from one network to an adjacent network is “forwarding”, over time the term “routing” has come to often be used both for a single network-to-network transfer as well as the overall process of transmitting a message from one device to another.

In the context of the OSI Reference Model, routing is an activity that generally takes place at the network layer—layer 3. Recall that data encapsulation causes a higher-layer message to be surrounded by headers and/or footers at the lower layers. When a message is routed, here's what happens:

- A high-level application on a machine decides to send a datagram to a distant computer. The datagram is packaged, and then passed down vertically through the protocol stack on the originating machine. Each layer encapsulates the data as described earlier. The datagram is addressed to the final destination device. When the message gets to the network layer and below, however, it is not packaged for local delivery directly to its ultimate destination, but rather to an *intermediate device*. This is the device that is responsible for routing to that destination network. The message is passed down to the data link layer and then the physical layer for transmission to that intermediate device.

- The intermediate device (often called a *router*) receives the message at the physical layer. It is passed up to the data link layer, where it is processed, checked for errors and so on, and the data link layer headers are removed. The resulting packet is passed up to the network layer. There, the intermediate device determines if the destination machine is on its local network, or if it needs to be forwarded to another intermediate device. It then repackages the message and passes it back *down* to the data link layer to be sent on the next leg of its journey.

- After several potential intermediate devices “handle” the message, it eventually reaches its destination. Here, it travels back up the protocol stack until it reaches the same layer as the one of the application that generated the message on the originating machine.

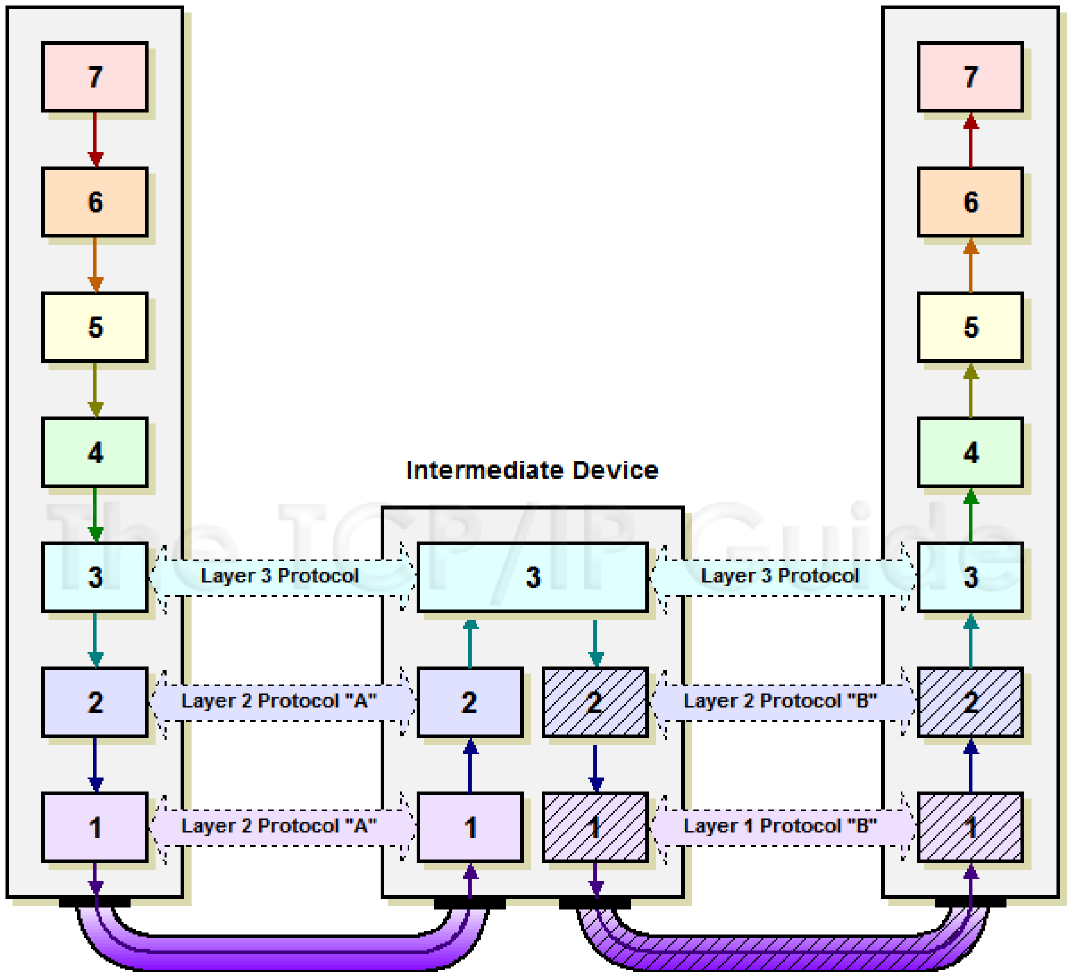

Figure 17: Message Routing in the OSI Reference Model

This diagram shows how routing is accomplished conceptually in the OSI model. The intermediate device connects the networks of the message transmitter and recipient. When data is sent, it is passed up to the network layer on the intermediate device, where it is repackaged and sent back down the stack for the next leg of its transmission. Note that the intermediate device actually has two different layer 1 and 2 implementations; one for the interface to each network. Also note that while the layer 3 protocol must be the same across the internetwork, each network can use different technologies at layers 1 and 2.

The key to this description is that in the intermediate devices, the message travels back up the OSI layers *only to the network layer*. It is then repackaged and sent back along its way. The higher layers are only involved on the source and destination devices. The protocol used at layer 3 must be common across the internetwork but each individual network can be different. This demonstrates some of the power of layering, by enabling even rather dissimilar physical networks to be connected together. The process is illustrated in Figure 17.

Key Concept: In the OSI model, the process of *routing* occurs when data is sent not directly from transmitter to ultimate recipient, but indirectly through the use of an intermediate system. That device, normally called a *router*, connects to two or more physical networks and thus has multiple interfaces to layer two. When it receives data, the data passes up only to the network layer, where it is repackaged and then sent on the next leg of its journey over the appropriate layer two interface.

TCP/IP Architecture and the TCP/IP Model

The OSI reference model consists of seven layers that represent a functional division of the tasks required to implement a network. It is a conceptual tool that I often use to show how various protocols and technologies fit together to implement networks. However, it's not the only networking model that attempts to divide tasks into layers and components. The TCP/IP protocol suite was in fact created before the OSI Reference Model; as such, its inventors didn't use the OSI model to explain TCP/IP architecture (even though the OSI model is often used in TCP/IP discussions today, as you will see in this Guide, believe me.)

The TCP/IP Model

The developers of the TCP/IP protocol suite created their own architectural model to help describe its components and functions. This model goes by different names, including the *TCP/IP model*, the *DARPA model* (after the agency that was largely responsible for developing TCP/IP) and the *DOD model* (after the United States Department of Defense, the “D” in “DARPA”). I just call it the TCP/IP model since this seems the simplest designation for modern times.

Regardless of the model you use to represent the function of a network—and regardless of what you call that model!—the functions that the model represents are pretty much the same. This means that the TCP/IP and the OSI models are really quite similar in nature even if they don't carve up the network functionality pie in precisely the same way. There is a fairly natural correspondence between the TCP/IP and OSI layers, it just isn't always a “one-to-one” relationship. Since the OSI model is used so widely, it is common to explain the TCP/IP architecture both in terms of the TCP/IP layers and the corresponding OSI layers, and that's what I will now do.

TCP/IP Model Layers

The TCP/IP model uses four layers that logically span the equivalent of the top six layers of the OSI reference model; this is shown in Figure 20. (The physical layer is not covered by the TCP/IP model because the data link layer is considered the point at which the interface occurs between the TCP/IP stack and the underlying networking hardware.) The following are the TCP/IP model layers, starting from the bottom.

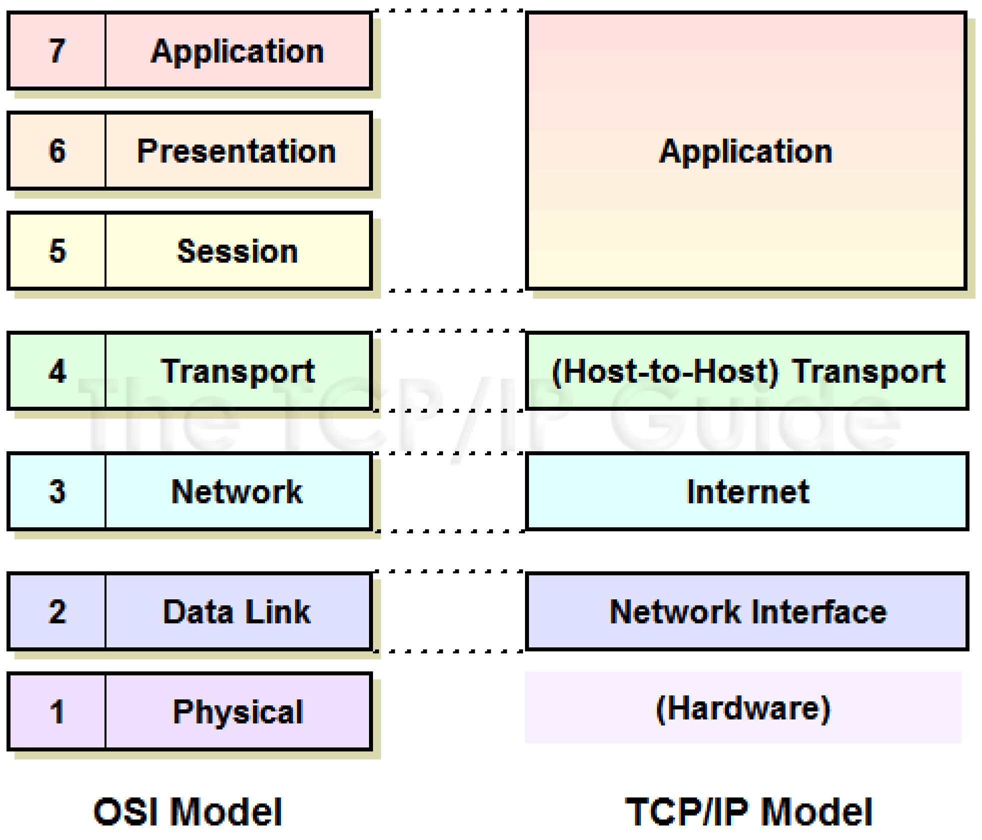

Figure 20: OSI Reference Model and TCP/IP Model Layers

The TCP/IP architectural model has four layers that approximately match six of the seven layers in the OSI Reference Model. The TCP/IP model does not address the physical layer, which is where hardware devices reside. The next three layers—*network interface*, *internet* and *(host-to-host) transport*—correspond to layers 2, 3 and 4 of the OSI model. The TCP/IP *application* layer conceptually "blurs" the top three OSI layers. It's also worth noting that some people consider certain aspects of the OSI session layer to be arguably part of the TCP/IP host-to-host transport layer.

Network Interface Layer

As its name suggests, this layer represents the place where the actual TCP/IP protocols running at higher layers interface to the local network. This layer is somewhat “controversial” in that some people don't even consider it a “legitimate” part of TCP/IP. This is usually because none of the core IP protocols run at this layer. Despite this, the network interface layer is part of the architecture. It is equivalent to the data link layer (layer two) in the OSI Reference Model and is also sometimes called the *link layer*. You may also see the name *network access layer*.

On many TCP/IP networks, there is no TCP/IP protocol running at all on this layer, because it is simply not needed. For example, if you run TCP/IP over an Ethernet, then Ethernet handles layer two (and layer one) functions. However, the TCP/IP standards do define protocols for TCP/IP networks that do not have their own layer two implementation. These protocols, the Serial Line Internet Protocol (SLIP) and the Point-to-Point Protocol (PPP), serve to fill the gap between the network layer and the physical layer. They are commonly used to facilitate TCP/IP over direct serial line connections (such as dial-up telephone networking) and other technologies that operate directly at the physical layer

Internet Layer

This layer corresponds to the network layer in the OSI Reference Model (and for that reason is sometimes called the *network layer* even in TCP/IP model discussions). It is responsible for typical layer three jobs, such as logical device addressing, data packaging, manipulation and delivery, and last but not least, routing. At this layer we find the Internet Protocol (IP), arguably the heart of TCP/IP, as well as support protocols such as ICMP and the routing protocols (RIP, OSFP, BGP, etc.) The new version of IP, called IP version 6, will be used for the Internet of the future and is of course also at this layer.

(Host-to-Host) Transport Layer

This primary job of this layer is to facilitate end-to-end communication over an internetwork. It is in charge of allowing logical connections to be made between devices to allow data to be sent either unreliably (with no guarantee that it gets there) or reliably (where the protocol keeps track of the data sent and received to make sure it arrives, and re-sends it if necessary). It is also here that identification of the specific source and destination application process is accomplished

The formal name of this layer is often shortened to just the *transport layer*; the key TCP/IP protocols at this layer are the Transmission Control Protocol (TCP) and User Datagram Protocol (UDP). The TCP/IP transport layer corresponds to the layer of the same name in the OSI model (layer four) but includes certain elements that are arguably part of the OSI session layer. For example, TCP establishes a connection that can persist for a long period of time, which some people say makes a TCP connection more like a session.

Application Layer

This is the highest layer in the TCP/IP model. It is a rather broad layer, encompassing layers five through seven in the OSI model. While this seems to represent a loss of detail compared to the OSI model, I think this is probably a good thing! The TCP/IP model better reflects the “blurry” nature of the divisions between the functions of the higher layers in the OSI model, which in practical terms often seem rather arbitrary. It really is hard to separate some protocols in terms of which of layers five, six or seven they encompass. (I didn't even bother to try in this Guide which is why the higher-level protocols are all in the same chapter, while layers one through four have their protocols listed separately.)

Numerous protocols reside at the application layer. These include application protocols such as HTTP, FTP and SMTP for providing end-user services, as well as administrative protocols like SNMP, DHCP and DNS.

Note: The internet and host-to-host transport layers are usually considered the “core” of TCP/IP architecture, since they contain most of the key protocols that implement TCP/IP internetworks.

In the topic that follows I provide a brief look at each of the TCP/IP protocols covered in detail in this Guide and more detail on where they all fit into the TCP/IP architecture. There I will also cover a couple of protocols that don't really fit into the TCP/IP layer model at all.

Key Concept: The architecture of the TCP/IP protocol suite is often described in terms of a layered reference model called the *TCP/IP model*, *DARPA model* or *DOD model*. The TCP/IP model includes four layers: the *network interface layer* (responsible for interfacing the suite to the physical hardware on which it runs), the *internet layer* (where device addressing, basic datagram communication and routing take place), the *host-to-host transport layer* (where connections are managed and reliable communication is ensured) and the *application layer* (where end-user applications and services reside.) The first three layers correspond to layers two through four of the OSI Reference Model respectively; the application layer is equivalent to OSI layers five to seven.