***Introduction to the OSI Model for Networking***

Networking models such as the OSI Reference Model provide a framework for breaking down complex internetworks into components that can more easily be understood and utilized. The model defines networking functions not as a large, complicated whole, but as a set of layered, modular components, each of which is responsible for a particular function. The result is better comprehension of network operations, improved performance and functionality, easier design and development, and the ability to combine different components in the way best suited to the needs of the network.

The OSI Reference Model helps you understand how networks and network protocols function. In the “real world”, it also helps you figure out which protocols and devices can interact with each other

While the model defines a framework for understanding networks, not all networking components, protocols and technologies will necessarily fall into the model’s strict layering architecture. There are cases where trying to use the model to describe certain concepts can lead to less clarity rather than more. One should remember that the OSI model is a tool, and should be used accordingly.

The most important OSI Reference Model concept is that of [networking](http://www.tcpipguide.com/free/t_OSIReferenceModelNetworkingLayersSublayersandLayer.htm) 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](http://www.tcpipguide.com/free/t_OSIReferenceModelNetworkingLayersSublayersandLayer.htm) used to implement a [network](http://www.tcpipguide.com/free/t_OSIReferenceModelNetworkingLayersSublayersandLayer.htm). The first and lowest layer is the physical layer, which is where low-level signalling 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](http://www.tcpipguide.com/free/t_OSIReferenceModelNetworkingLayersSublayersandLayer.htm).

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](http://www.tcpipguide.com/free/t_OSIReferenceModelNetworkingLayersSublayersandLayer.htm).

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](http://www.tcpipguide.com/free/t_OSIReferenceModelNetworkingLayersSublayersandLayer.htm) to another. It is the domain of hardware engineers and signalling experts. The second layer is a bit more abstract but still deals with signalling and hardware. As you proceed through the third, fourth and subsequent layers, the [technologies](http://www.tcpipguide.com/free/t_OSIReferenceModelNetworkingLayersSublayersandLayer.htm) 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](http://www.tcpipguide.com/free/t_OSIReferenceModelNetworkingLayersSublayersandLayer.htm) 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: (See Figure 1)

Lower Layers (Layers 1, 2, 3 and 4): The lower layers of the model—physical, data link, network and [transport](http://www.tcpipguide.com/free/t_OSIReferenceModelNetworkingLayersSublayersandLayer-2.htm)—are primarily concerned with the formatting, encoding and [transmission](http://www.tcpipguide.com/free/t_OSIReferenceModelNetworkingLayersSublayersandLayer-2.htm) 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](http://www.tcpipguide.com/free/t_OSIReferenceModelNetworkingLayersSublayersandLayer-2.htm), 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](http://www.tcpipguide.com/free/t_OSIReferenceModelNetworkingLayersSublayersandLayer-2.htm) of data. These layers are almost always implemented as software running on a [computer](http://www.tcpipguide.com/free/t_OSIReferenceModelNetworkingLayersSublayersandLayer-2.htm) or other hardware device. 

|  |
| --- |
| **http://www.tcpipguide.com/free/diagrams/osilayers.png** |
| Figure 1 |

***Vertical Communication***

|  |
| --- |
| See Figure 2. Vertical communication is done up and down the protocol stack every time anything is sent across the network, and of course, whenever anything is received. This occurs because the higher levels are implemented as logical functions, in software; there is no actual physical connection. The higher layers package data and send it down to the lower layers for it to be sent across the network. At the very lowest level, the data is sent over the network. On the receiving end, the process is reversed, with the data traveling back up to the higher layers on the receiving device. [The next topic dealing with horizontal communication](http://www.tcpipguide.com/free/t_ProtocolsHorizontalCorrespondingLayerCommunication.htm) explains more about this logical interaction between corresponding layers |

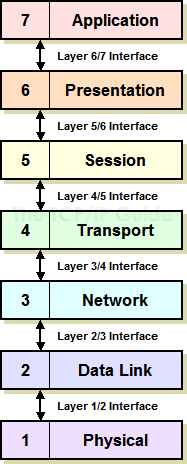


Figure 2

Many different types of communication actually take place between layers through the interfaces. Control [information](http://www.tcpipguide.com/free/t_InterfacesVerticalAdjacentLayerCommunication-2.htm) is passed to enable the higher layers to utilize the lower ones, and for the lower ones to pass status and results information back to the higher ones. [Data](http://www.tcpipguide.com/free/t_InterfacesVerticalAdjacentLayerCommunication-2.htm) is also passed in both directions across the interface. For transmission, it flows down to the lower layer, each time normally resulting in [data encapsulation](http://www.tcpipguide.com/free/t_DataEncapsulationProtocolDataUnitsPDUsandServiceDa.htm). Upon reception, the process is reversed, with data being sent back up across the interface from lower to higher layer.

In summary then:

An interface defines the mechanism for vertical communication between adjacent layers. The existence of well-defined interfaces between layers is what permits a higher layer to use the services of any of a number of lower layers, without requiring knowledge of how those layers are implemented.

***Horizontal (Corresponding Layer) Communication*** 

Let's consider how these corresponding layers communicate using protocols. First, recall that every layer in the model, except the bottom (physical) layer, is really a program or algorithm running on a computer. There is no way for, say, a Web browser and a [Web server](http://www.tcpipguide.com/free/t_ProtocolsHorizontalCorrespondingLayerCommunication-2.htm) to actually connect together directly—they are just [software programs](http://www.tcpipguide.com/free/t_ProtocolsHorizontalCorrespondingLayerCommunication-2.htm), after all. Instead, the software running at various layers communicates logically. That is to say, through the use of software and procedures, a process running at layer 5 on one machine can accomplish logical [communication](http://www.tcpipguide.com/free/t_ProtocolsHorizontalCorrespondingLayerCommunication-2.htm) with a similar process running at layer 5 on another machine.

Since machines are only physically connected at layer 1, this means that in order for a protocol at layer 5 to function, the data on the sending machine must “pass down” the data through the layers between layer 5 and layer 1. The data is then transmitted over the physical connection to layer 1 of the other machine, and “passed up” the protocol stack of the receiving machine to layer 5. This is how the two machines are logically linked at layer 5, even though they have no physical connection at that layer.

Thus, with the exception of the actual physical connection at layer 1, all horizontal communication also requires vertical communication—down the stack on one machine, and then back up the stack on the other. This process is illustrated in [Figure 3](http://www.tcpipguide.com/free/t_ProtocolsHorizontalCorrespondingLayerCommunication-2.htm#Figure_14). (The communication doesn’t always go all the way back up the stack for each connection, however, as in the case of [routing](http://www.tcpipguide.com/free/t_IndirectDeviceConnectionandMessageRouting.htm).)

|  |
| --- |
| http://www.tcpipguide.com/free/diagrams/osiprotocols.png |
| Figure 3 |

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

Protocols are what describe the rules that control [horizontal communication](http://www.tcpipguide.com/free/t_ProtocolsHorizontalCorrespondingLayerCommunication.htm), that is, conversations between processes that run at corresponding layers within the [OSI](http://www.tcpipguide.com/free/t_DataEncapsulationProtocolDataUnitsPDUsandServiceDa.htm) Reference Model. At every layer (except layer one) these [communications](http://www.tcpipguide.com/free/t_DataEncapsulationProtocolDataUnitsPDUsandServiceDa.htm) ultimately take the form of some sort of message that is sent between corresponding software elements on two or [more devices](http://www.tcpipguide.com/free/t_DataEncapsulationProtocolDataUnitsPDUsandServiceDa.htm). 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](http://www.tcpipguide.com/free/t_ProtocolsHorizontalCorrespondingLayerCommunication.htm), 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](http://www.tcpipguide.com/free/t_DataEncapsulationProtocolDataUnitsPDUsandServiceDa.htm). We’ve also already seen that [using OSI terminology](http://www.tcpipguide.com/free/t_NNotationandOtherOSIModelLayerTerminology.htm), 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](http://www.tcpipguide.com/free/t_DataEncapsulationProtocolDataUnitsPDUsandServiceDa.htm) 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 4](http://www.tcpipguide.com/free/t_DataEncapsulationProtocolDataUnitsPDUsandServiceDa.htm#Figure_15).

|  |
| --- |
|  |

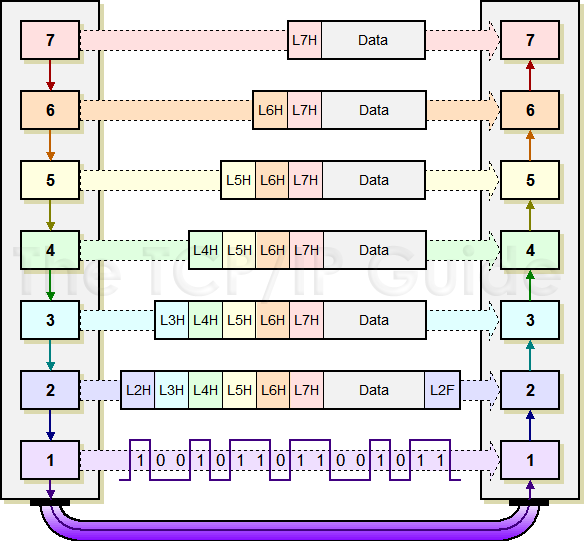


Figure 4

***PDU and SDU***

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](http://www.tcpipguide.com/free/t_DataEncapsulationProtocolDataUnitsPDUsandServiceDa-3.htm). In summary: a layer N PDU is a layer N-1 SDU, which is encapsulated into a layer N-1 PDU.

***Routing and the OSI Model***

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](http://www.tcpipguide.com/free/t_IndirectDeviceConnectionandMessageRouting.htm) 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.

***Understanding The OSI Reference Model: An Analogy***

I have attempted in this discussion of the OSI Reference Model to provide as much “plain English” explanation of how it works as I could. However, there are situations where a good analogy can accomplish what lots of descriptions cannot. So, I am going to attempt to illustrate [the key](http://www.tcpipguide.com/free/t_UnderstandingTheOSIReferenceModelAnAnalogy.htm) OSI model concepts ([layers](http://www.tcpipguide.com/free/t_OSIReferenceModelNetworkingLayersSublayersandLayer.htm), [vertical communication](http://www.tcpipguide.com/free/t_InterfacesVerticalAdjacentLayerCommunication.htm), [horizontal communication](http://www.tcpipguide.com/free/t_ProtocolsHorizontalCorrespondingLayerCommunication.htm), [data encapsulation](http://www.tcpipguide.com/free/t_DataEncapsulationProtocolDataUnitsPDUsandServiceDa.htm) and [message routing](http://www.tcpipguide.com/free/t_IndirectDeviceConnectionandMessageRouting.htm)) by way of a real-life analogy. You can be the judge of whether it is a good analogy or not. Just remember that no analogy is perfect.

Our scenario seems relatively simple and common: the CEO of a Fortune 500 [company](http://www.tcpipguide.com/free/t_UnderstandingTheOSIReferenceModelAnAnalogy.htm) needs to send a letter to the CEO of another large company. Simple, right? Just like firing up your browser and connecting to your favourite [Web site](http://www.tcpipguide.com/free/t_UnderstandingTheOSIReferenceModelAnAnalogy.htm) is simple. However, in both cases, a lot goes on “behind the scenes” to make the communication happen. In the analogy shown in [Table 1](http://www.tcpipguide.com/free/t_UnderstandingTheOSIReferenceModelAnAnalogy.htm#Table_17) below, I compare these real-world and "cyber-world” communications.

|  |  |  |  |
| --- | --- | --- | --- |
| **Table 1: OSI Reference Model Real-World Analogy** | | | |
| **Phase** | **OSI Layer** | **CEO Letter** | **Web Site Connection (Simplified)** |
| **Transmission** | **7** | The CEO of a company in Phoenix decides he needs to send a letter to a peer of his in Albany. He dictates the letter to his administrative assistant. | You decide you want to connect to the [web server](http://www.tcpipguide.com/free/t_UnderstandingTheOSIReferenceModelAnAnalogy.htm) at IP address 10.0.12.34, which is within your organization but not on your local network. You type the address into your browser. |
| **6** | The administrative assistant transcribes the dictation into writing. | (Generally, with a web site connection, nothing happens at this layer, but format translation may be done in some cases.) |
| **5** | The administrative assistant puts the letter in an envelope and gives it to the mail room. The assistant doesn't actually know how the letter will be sent, but he knows it is urgent so he says, “get this to its destination quickly”. | The request is sent via a call to an [application program](http://www.tcpipguide.com/free/t_UnderstandingTheOSIReferenceModelAnAnalogy.htm) interface (API), to issue the command necessary to contact the server at that address. |
| **4** | The mail room must decide how to get the letter where it needs to go. Since it is a rush, the people in the mail room decide they must use a courier. The envelope is given to the courier company to send. | The Transmission Control Protocol (TCP) is used to create a segment to be sent to IP address 10.0.12.34. |
| **Routing** | **3** | The courier company receives the envelope, but it needs to add its own handling information, so it places the smaller envelope in a courier envelope (encapsulation). The courier then consults its airplane route information and determines that to get this envelope to Albany, it must be flown through its hub in Chicago. It hands this envelope to the workers who load packages on airplanes. | [Your computer](http://www.tcpipguide.com/free/t_UnderstandingTheOSIReferenceModelAnAnalogy.htm) creates an IP datagram encapsulating the TCP datagram created above. It then addresses the packet to 10.0.12.34. but discovers that it is not on its local network. So instead, it realizes it needs to send the message to its designated routing device at IP address 10.0.43.21. It hands the packet to the driver for your [Ethernet](http://www.tcpipguide.com/free/t_UnderstandingTheOSIReferenceModelAnAnalogy.htm) card (the software that [interfaces](http://www.tcpipguide.com/free/t_UnderstandingTheOSIReferenceModelAnAnalogy.htm) to the Ethernet hardware). |
| **2** | The workers take the courier envelope and put on it a tag with the code for Chicago. They then put it in a handling box and then load it on the plane to Chicago. | The Ethernet card driver forms a frame containing the IP datagram and prepares it to be sent over the network. It packages the message and puts the address 10.0.43.21 (for the router) in the frame. |
| **1** | The plane flies to Chicago. | The frame is sent over the twisted pair cable that connects your local area network. (I'm ignoring overhead, collisions, etc. here, but then I also ignored the possibility of collisions with the plane.) |
| **2** | In Chicago, the box is unloaded, and the courier envelope is removed from it and given to the people who handle routing in Chicago. | The Ethernet card at the machine with IP address 10.0.43.21 receives the frame, strips off the frame headers and hands it up to the network layer. |
| **3** | The tag marked “Chicago” is removed from the outside of the courier envelope. The envelope is then given back to the airplane workers to be sent to Albany. | The IP datagram is processed by the router, which realizes the destination (10.0.12.34) can be reached directly. It passes the datagram back down to the Ethernet driver. |
| **2** | The envelope is given a new tag with the code for Albany, placed in another box and loaded on the plane to Albany. | The Ethernet driver creates a new frame and prepares to send it to the device that uses IP address 10.0.12.34. |
| **1** | The plane flies to Albany. | The frame is sent over the network. |
| **2** | The box is unloaded and the courier envelope is removed from the box. It is given to the Albany routing office. | The Ethernet card at the device with IP address 10.0.12.34 receives the frame, strips off the headers and passes it up the stack. |
| **Reception** | **3** | The courier company in Albany sees that the destination is in Albany, and delivers the envelope to the destination CEO's company. | The IP headers are removed from the datagram and the TCP segment handed up to TCP. |
| **4** | The mail room removes the inner envelope from the courier envelope and delivers it to the destination CEO's assistant. | TCP removes its headers and hands the data up to the [drivers](http://www.tcpipguide.com/free/t_UnderstandingTheOSIReferenceModelAnAnalogy.htm)on the destination machine. |
| **5** | The assistant takes the letter out of the envelope. | The request is sent to the Web [server software](http://www.tcpipguide.com/free/t_UnderstandingTheOSIReferenceModelAnAnalogy.htm) for processing. |
| **6** | The assistant reads the letter and decides whether to give the letter to the CEO, transcribe it to email, call the CEO on her cell phone, or whatever. | (Again, in this example nothing probably happens at the Presentation layer.) |
| **7** | The second CEO receives the message that was sent by the first one. | The Web server receives and processes the request. |

As you can see, the processes have a fair bit in common. The vertical communication and encapsulation are pretty obvious, as is the routing. Also implied is the horizontal communication that occurs logically—the two CEOs seem to be “connected” despite all that happens to enable this to occur. Similarly, the two assistants are logically connected as well, in a way, even though they never actually converse. Of course, this example is highly simplified in just about every way imaginable, so please don’t use it as a way of trying to learn about how TCP/IP works

***Layer 1: 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 Signalling: The physical layer is responsible for various encoding and signalling 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](http://www.tcpipguide.com/free/t_PhysicalLayerLayer1.htm), 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](http://www.tcpipguide.com/free/t_PhysicalLayerLayer1.htm) 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.

***Layer 2: 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 sub-layer; it provides services to [the network](http://www.tcpipguide.com/free/t_DataLinkLayerLayer2.htm) 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](http://www.tcpipguide.com/free/t_DataLinkLayerLayer2.htm) use a shared medium (such as a single [network cable](http://www.tcpipguide.com/free/t_DataLinkLayerLayer2.htm), 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](http://www.tcpipguide.com/free/t_DataLinkLayerLayer2.htm)/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 thephysical 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.

***Layer 3: 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](http://www.tcpipguide.com/free/t_NetworkLayerLayer3.htm) 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](http://www.tcpipguide.com/free/t_NetworkLayerLayer3.htm) 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](http://www.tcpipguide.com/free/t_IndirectDeviceConnectionandMessageRouting.htm), and show how [it works](http://www.tcpipguide.com/free/t_NetworkLayerLayer3.htm) by way of an [OSI model analogy](http://www.tcpipguide.com/free/t_UnderstandingTheOSIReferenceModelAnAnalogy.htm).
* Datagram Encapsulation: The network layer normally [encapsulates](http://www.tcpipguide.com/free/t_DataEncapsulationProtocolDataUnitsPDUsandServiceDa.htm) 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](http://www.tcpipguide.com/free/t_NetworkLayerLayer3.htm) 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](http://www.tcpipguide.com/free/t_IPDatagramSizeMaximumTransmissionUnitMTUFragmentat.htm).
* 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](http://www.tcpipguide.com/free/t_NetworkLayerLayer3.htm) about the status of hosts on the network or the devices themselves.

***Layer 4: 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](http://www.tcpipguide.com/free/t_TransportLayerLayer4-2.htm) on a local [network](http://www.tcpipguide.com/free/t_TransportLayerLayer4-2.htm), and layer three addressing identifies [devices](http://www.tcpipguide.com/free/t_TransportLayerLayer4-2.htm)on a logical internetwork. Addressing is also performed at the transport layer, where it is used to differentiate between [software programs](http://www.tcpipguide.com/free/t_TransportLayerLayer4-2.htm). 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](http://www.tcpipguide.com/free/t_TCPIPTransportLayerProtocolTCPandUDPAddressingPort.htm) 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](http://www.tcpipguide.com/free/t_TransportLayerLayer4-2.htm) received from many application programs for transport, combining them into a single stream of data to be sent. The same protocols receive data and thendemultiplex 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](http://www.tcpipguide.com/free/t_TransportLayerLayer4-2.htm) 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](http://www.tcpipguide.com/free/t_TransportLayerLayer4-2.htm) 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.

***Layer 5: Session Layer Functions***

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](http://www.tcpipguide.com/free/t_SessionLayerLayer5-2.htm) 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)](http://www.tcpipguide.com/free/t_TCPIPTransmissionControlProtocolTCP.htm) 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](http://www.tcpipguide.com/free/t_TCPIPArchitectureandtheTCPIPModel.htm) really straddles OSI layers four and five).

***Layer 6: 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](http://www.tcpipguide.com/free/t_PresentationLayerLayer6.htm). 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](http://www.tcpipguide.com/free/t_PresentationLayerLayer6.htm) (and decryption) are performed at the presentation layer. This ensures the security of the data as it travels down the [protocol stack](http://www.tcpipguide.com/free/t_PresentationLayerLayer6.htm). For example, one of the most popular [encryption schemes](http://www.tcpipguide.com/free/t_PresentationLayerLayer6.htm) 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](http://www.tcpipguide.com/free/t_IPSecurityIPSecProtocols.htm).

***Layer 7: Application Layer Functions***

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](http://www.tcpipguide.com/free/t_ApplicationLayerLayer7.htm) applications. These programs are what actually implement the functions performed by users to accomplish various tasks over [the network](http://www.tcpipguide.com/free/t_ApplicationLayerLayer7.htm).

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](http://www.tcpipguide.com/free/t_ApplicationLayerLayer7.htm) to employ. For example, when you use your [Web browser](http://www.tcpipguide.com/free/t_ApplicationLayerLayer7.htm), that actual software is an application running on [your PC](http://www.tcpipguide.com/free/t_ApplicationLayerLayer7.htm). 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)](http://www.tcpipguide.com/free/t_TCPIPHypertextTransferProtocolHTTP.htm). 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](http://www.tcpipguide.com/free/t_ApplicationLayerLayer7.htm) 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](http://www.tcpipguide.com/free/t_ApplicationLayerLayer7.htm) 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](http://www.tcpipguide.com/free/t_ApplicationLayerLayer7.htm) 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](http://www.tcpipguide.com/free/t_ApplicationLayerLayer7.htm), NFS, Telnet, SNMP, POP3, NNTP and IRC. Lots of alphabet soup, sorry. J I describe all of these and more in [the chapter on higher-layer protocols and applications](http://www.tcpipguide.com/free/t_UpperLayerLayers56and7NetworkingProtocolsServicesa.htm).

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.