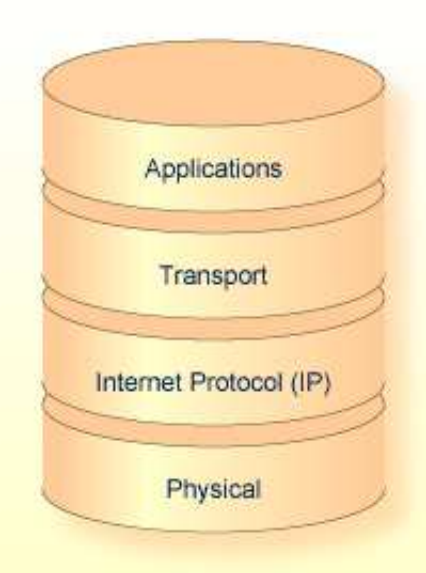
TCP/IP Protocol Stack

As shown in the following diagram, the TCP/IP protocol stack contains four layers:

- Physical layer
- Internet Protocol (IP) layer
- Transport layer, comprising
  - Transmission Control Protocol (TCP); and
  - User Datagram Protocol (UDP)
- Applications layer



This chapter covers the following topics:

- Physical Layer
- Internet Protocol (IP) Layer
- Transport Layer
- Applications Layer

Physical Layer

At the bottom of the stack is the physical layer, which deals with the actual transmission of data over physical media such as serial lines, Ethernet, token rings, FDDI rings, and hyperchannels. Messages can also be sent and received over other, non-physical access methods such as VTAM/SNA.

Internet Protocol (IP) Layer

Above the physical layer is the Internet Protocol (IP) layer, which deals with the routing of packets from one computer to another. The IP layer

- determines which lower level protocol to use when multiple interfaces exist.
- determines whether to send a packet directly to the host or indirectly to a relay host known as a router.

  When a packet is larger than the size supported by the physical medium, the IP layer breaks the packet into smaller packets, a process referred to as "fragmentation and reassembly".
- provides some control services for packets, and ensures that they are not sent from router to router indefinitely.

However, the IP layer does not keep track of a packet after it is sent, nor does it guarantee that the packet will be delivered.

Transport Layer

Above the IP layer is the transport layer, which contains the Transmission Control Protocol (TCP) and the User Datagram Protocol (UDP).

Transmission Control Protocol (TCP)

The Transmission Control Protocol (TCP) guarantees that data sent by higher levels is delivered in order and without corruption. To accomplish this level of service, the TCP implementation on one computer establishes a session or connection with the TCP implementation on another computer. This process is referred to as Connection Oriented Transport Service (COTS).

After a session is established, data is sent and received as a stream of contiguous bytes; each byte can be referenced by an exact sequence number. When data is received by the remote TCP, it sends an acknowledgment back to the local TCP advising it of the sequence number of the last byte of data received. If an acknowledgment is not received, or if an acknowledgment for previously sent data is received twice, the local TCP retransmits the data until it is all acknowledged. The remote TCP discards any bytes that are received more than once.

All data sent and received by TCP is validated for corruption using checksums. Whenever a checksum is incorrect, the bad data is discarded by TCP, and the correct data is retransmitted until it is accurately received.

User Datagram Protocol (UDP)

Unlike the TCP, the User Datagram Protocol (UDP) transmits and receives data in packets (datagrams), and delivery is not guaranteed. The contents of the data can be sent with or without a checksum. The use of checksums varies widely from one implementation to another.

Applications Layer

Above the transport layer is the applications layer, which contains both general applications and function libraries for use by applications.

Some general applications that run over TCP include

- File Transfer Protocol (FTP);
- remote terminal emulation (TELNET in line mode, TN3270 in full screen);
- Electronic Mail (SMTP); and
- Entire Net-Work.

  Some general applications that run over UDP are

- Network File Server (NFS); and
- Domain Name Server (DNS).

Interface with TCP and UDP

Function libraries provide routines to simplify the interface between applications and TCP/UDP of the Transport layer:

- The most common function library is known as Sockets, which allows an application written in C to access TCP as if it were just another stream input/output device.
- Another function library that is less commonly used is Remote Procedure Call (RPC), which allows applications to make calls to functions that are located in another application on a different computer.

  The environment in which an application runs often dictates the interface used between it and TCP or UDP:

- Most UNIX, OS/2, and Windows applications are written in C and utilize a direct socket interface.
- On IBM mainframes and other systems based on the same architecture such as Fujitsu Technology Solutions, applications are often written in S/390 assembler, and use either a pseudo-socket interface or an application program interface (API) to gain access to the TCP/IP protocol stack.

Ports

The interface that exists between an application and TCP is referred to as a port. Ports are classified as server ports and client ports:

- Server ports are generally ports on which the application "listens" for incoming connections to be made.
- Client ports are generally ports on which the application "connects" outwardly to a server port.

An application may control multiple client ports and server ports simultaneously.

Each port is identified by a port number, which ranges from 1 to 65535.

- The port number used by client ports usually has no significance and is often assigned by TCP.
- Server port numbers, however, are usually required to be "well known"; that is, the client must know which port the server is listening on when it attempts to connect. Server port numbers usually are specified by the server application.