Transport layer provides a logical presentation between application processes running on different hosts. Logical means it is as if hosts running the processes were directly connected but in reality the hosts may be far apart only connected via numerous routers and a range of links.
Application processes use logical communication provided by the transport layer to send messages to each other without worrying of the details of the physical infrastructure used to carry this messages.
Therefore
Transport layer protocols are implemented in end systems but not in network routers.
On the sending side, the transport layer converts the application layer messages (a letter put in an envelope and addressed) it receives from a sending application process into transport layer packets known as segments.
This is done by breaking the application messages into smaller chunks and adding a transport layer header to each chunk to create the transport layer segment. It is important to note that network routers act only on the network layer fields of the datagram i.e they do not examine the fields of the transport layer segment encapsulated within the datagram
On the receiving side, the network layer extracts the transport layer segment from the datagram and passes it up to the transport layer which then processes the received segment making the data in the segment available to the receiving application.

There are two major protocols; TCP and UDP. Each of them provides different sets of transport layer services to the specific application. TCP provides reliable data transfer. It uses flow control, sequence numbers, acknowledgement and timers to ensure reliable data transfers and that the packets are received in order.

Multiplexing and Demultiplexing

Putting together into a bunch — Multiplexing

putting one by one from a bunch — De-Multiplexing

Principles of Reliable data transfer in TCP

One of, these aspects that ensure reliable data transfer is the use of both positive acknowledgement (OK) and negative acknowledgements (please repeat that). These control messages allow the receiver to let the sender know what has been received correctly and what has been received in error and requires repeating. In a computer network setting. Reliable data transfer protocols based on such retransmissions are known as automatic repeat request (ARQ). Therefore, three additional protocol abilities are required in ARQ protocols to handle the presence of bit errors.

A) Error Detection

Error detection and correction techniques allow the receiver to detect and possibly correct packet bit

Reliable Data Transfer

errors

B) Receiver Feedback

The positive (ACK) is called and a negative (NAK) (Negative acknowledgement) are examples of such feedback

9 ReTransmissions

for a package received in error

TCP is said to be Connection oriented because before an application process can begin to send data to another, the two processes must unshake i.e must send some preliminary segment which allow to establish the parameters of ensuring reliable data transfer. This occurs in a 3-way handshake by the first client sending a special TCP segment to check if the server is available. The server responds with another TCP segment to confirm its availability and ready to connect. The client responds with the final special segments. The first two segments carry no payload (data). The third of these segments may carry a payload.

b) Flow control

When TCP connection receives bytes that are correct and in sequence. It places the data in the received buffer. The associated application will read the data from this buffer. If the application is relatively slow at reading the data. The overflow the connection sender can easily

received buffer by sending too much data too quickly.
TCP flow control service ensures that the sender does not overflow the receiver. Flow control is therefore a speed matching service between the sending rate and the receiving rate
How?
TCP does this by maintaining a variable called the receive window by the sender It gives the sender an idea of how much free buffer space is available at the receiver.
Example
Host A is sending a large file to host B and host B allocates a received buffer to the connection called ~~RCV~~ RcvBuffer from time to time the application process in host B reads from the Buffers. and the following variables are defined
a) LastByteRead - The number of the last byte in the data stream read from the buffer by the application process in B
b) LastByteRcvd - number of the last byte in the data stream that has arrived from the network and had been placed in the received buffer at B.

3 ~~In TCP~~ Congestion Control
It is used by GCP to have each sender to limit the rate at which it sends traffic into its connection depending on the Network congestion level.

If a sender perceives that there is little congestion ~~connection~~ on the path then TCP sender increases its send rate and v.ce versa, Direct ~~feedback~~ may be sent from a network router to a sender using a ~~that~~ choke packet (congested)
Second form is a router can mark/update a field ~~in a packet that is~~ flowing from sender to receiver to indicate congestion.

TCP congestion control mechanism maintains a number of variables such as a receive buffer, a send buffer, a ~~congestion window~~ and other additional variables to check for congestion. The congestion window (CWND) imposes a constraint on the rate at which a TCP sender can send traffic into the network i.e., the amount of unacknowledged ~~data~~ that at a sender may not exceed the minimum of ~~the~~ control window (CWND) and receive window (RWND) ~~it~~ i.e., ~~last byte~~ ~~lastByteSent~~

LastByteSent - LastByteAcked $\leq$ min {cwnd, rwnd}

10 - 6 $\leq$ min {7, 6}

UDP

Is a connectionless protocol. It takes messages from the application process. Attaches source and destination port numbers and passes the resulting segment to the network layer. At the segment arrived at the receiving port. UDP uses the destination port number to

deliver the segment data to the correct application process. Note that with UDP that there is no handshaking between sending and receiving transport layer entities before sending transport layer segments. Therefore UDP is said to be Connectionless.

Now, Why would an application developer choose to build an application over UDP than TCP?

A Final Application Level control over what data is sent and when.

Under UDP, as soon as an application process passes data to UDP, UDP will package the data inside a UDP segment and immediately pass the segment to the network layer. TCP on the other hand has congestion control mechanisms that throttles the transport layer TCP sender when one or more links become excessively congested. TCP will also continue to resend segments until the receipt of the segments is acknowledged by the receiver.

For application that do not require such TCP requirements are not necessary and therefore UDP works better.

② No Connection Establishment

UDP starts sending without any formal preliminaries like the 3-way handshake therefore UDP does not introduce any delay to establish a connection. That is the reason why DNS runs over UDP and not TCP.

3) No connection state

TCP maintains connection state in end systems including receive and send buffers sequence and acknowledgement numbers, congestion control parameters etc
UDP on the other hand does not track this parameters
for this reason, a server devoted to a particular application can typically support many more active clients when the application runs on UDP rather than TCP

~~Small packet Header~~ Overhead

The TCP segments has 20 bytes of header overhead in every segment whereas UDP has only 8 bytes of overhead.

Examples of

Application	Application layer protocol	Transport ~~App~~ protocol used
eMail	SMTP	TCP
Remote Terminal Access	Telnet	TCP
Web (HTTP)	HTTP	TCP
File transfer	FTP	TCP
Remote file server	NFS	UDP
Streaming Multimedia	Depends on developer	UDP/TCP
Internet telephony	Depend on dev	UDP/TCP
Network management	SNMP	UDP
Routing protocol	RIP	UDP
Name translation	DNS	UDP