# UNIT – IV

# TRANSPORT LAYER

**INTRODUCTION:**

* The Transport layer in the TCP/IP suite is located between the application layer and the network layer.
* It provides service to the application layer and receives services from the network layer.
* The transport layer acts as a liaison between a client program and a server program.
* The transport layer is the heart of the TCP/IP protocol suite.
* It is the end-to-end logical vehicle for transferring data from one point to another in the internet.

**TRANSPORT LAYER PROTOCOL**

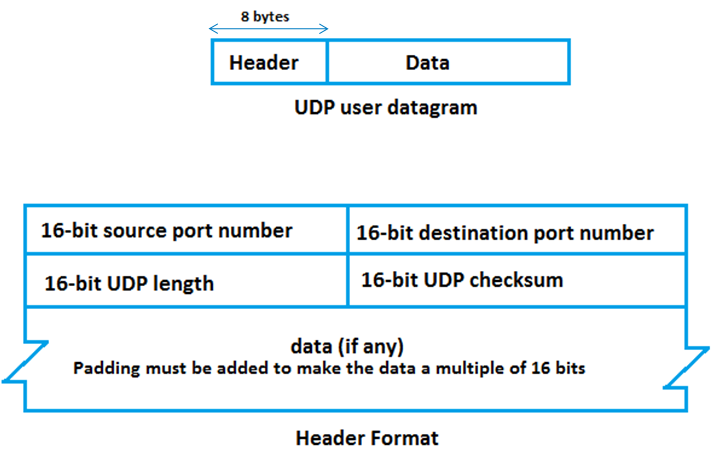
* Simple protocol
* Stop and wait protocol
* Go- back- n protocol
* Selective –repeat protocol.

**USER DATAGRAM PROTOCOL**

* UDP - User datagram protocol.
* The UDP is a connection less, unreliable transport protocol.
* UDP is a very simple protocol using a minimum of overhead.

**User datagram:**

* UDP packets are called user Datagram.
* It has a fixed size header of 8 bytes made of our fields, each of 2 bytes (16 bits).



* The first two fields define the source and destination port numbers, the third field defines the total length of the user datagram, header plus data.
* The last field can carry the optional checksum.

**UDP services:**

* Process–to-process communication.
* Connection less service
* Flow control
* Error control
* Checksum
* Congestion control
* Encapsulation and decapsulation
* Multiplexing and demultiplexing.

**Process –to-process communication:**

* UDP provides process to process communication using socket address, a combination of IP address and port numbers.

**Connection less service:**

* UDP provides a connection less service this means that each user datagram sent by UDP is an independent datagram.
* There is no relationship between the different user datagrams even if they are coming from the same source process and going to the same destination program.
* The user datagrams are not numbered, also unlike TCP there is no connection establishment and no connection termination, this means that each user datagram can travel on a different path.

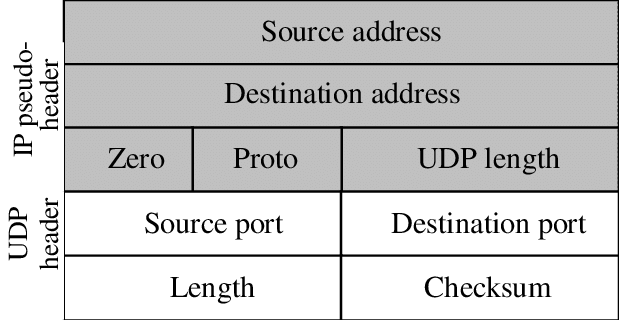
**Flow control:**

* UDP is a very simple protocol.
* There is no flow control and hence no windows mechanism.

**Error control:**

* There is no error control mechanism in UDP except for the checksum. This means that the sender does not know if a message has been lost or duplicated.
* When the receiver detects an error through the checksum the user datagram is silently discarded.

**Checksum :**



* UDP checksum calculation includes three sections.

1. A pseudo header
2. UDP header
3. Data.

* The pseudo header is the part of the header of the IP packet.
* The header include source and destination port address.
* The data coming from the application layer

**Congestion control:**

* UDP is a connection less protocol, if does not provides congestion control.

**Encapsulation and Decapsulation:**

* To send a message from one process to another the UDP protocol encapsulation and decapsulation messages.

**Multiplexing and demultiplexing:**

* In a host running a TCP/IP protocol suite, there is only one UDP but possibly several process that may want to use the services of UDP, to handle this situation UDP multiplexies and demultiplexes.

**TRANSMISSION CONTROL PROTOCOL**

**Introduction:**

* TCP is a connection oriented, reliable protocol.
* TCP explicitly defines connection establishment, data transfer and connection tear-down phases to provide a connection oriented service.
* TCP uses a combination of GBN and SR protocols to provides reliability.
* To achieve this goal, TCP Uses checksum, retransmission of the lost or corrupted packets, cumulative and selective acknowledgment and timers.
* TCP is the most common transport layer protocol in the internet.

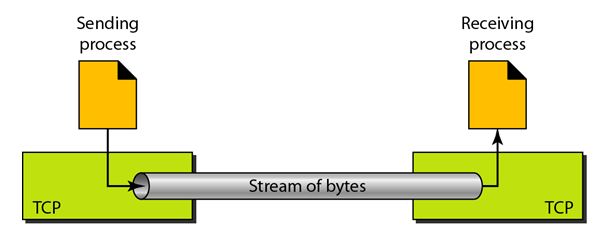
**TCP SERVICES**

* Process to process communications
* Stream delivery service.
* Sending and receiving buffers
* Segments
* Full duplex communications
* Multiplexing and demultiplexing
* Connection oriented service
* Reliable service

**Process to process communication:**

* TCP provides process to process communication using port numbers.

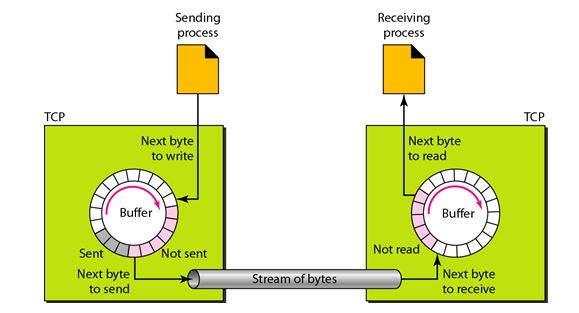
**Stream delivery service:**

****

* TCP is a stream oriented protocol.
* TCP allows the sending process to deliver data as a stream of bytes and allows the receiving process to obtain data as a stream of bytes.
* TCP creates an environment in which the two processes seem to be connected by an imaginary “tube” that carries their bytes across the internet.

**Sending and receiving buffers:**

* TCP needs buffer for storage;
* There are two buffers, the sending buffer the receiving buffer.
* Way to implement a buffer is to use a circular array of 1 byte locations.
* We have show two buffers of 20 bytes each, normally the buffers are hundreds or thousands of bytes, depending an the implementation.
* The diagram shows the movement of the data in one direction, at the sender the buffers has three types of champers.
* The white section contains empty champers that can be filled by the sending process.
* The colored area holds bytes that have been sent but not yet acknowledged.



* The shaded area contains bytes to be sent by the sending TCP.
* The operation of the buffer at the receiver is simpler,
* The circular buffer is divided into two areas,
* The white area contains empty chambers to be filled by bytes received from the network.
* The colored sections contain received bytes that can be read by the receiving process.
* When a byte is read by the receiving process, the chamber is recycled and added to the pool of empty chambers.

**Segments:**

* The network layer, as a service provider for TCP needs to send data in packets not as a stream of bytes, at the transport layer, TCP groups a number of bytes together into a packet called a segment.
* TCP adds a header to each segment and delivers the segment to the network layer for transmission

**Full duplex communication:**

* TCP offers full duplex service, where data can flow in both direction at the same time.

**Multiplexing and demultiplexing:**

* TCP performs multiplexing at the sender and demultiplexing at the receiver.

**Connection oriented service:**

* TCP is a connection oriented protocol
* The following three phases are,
* The two TCP’s establish a logical connection between them.
* Data are exchanged in both direction.
* The connection is terminated.

**Reliable service:**

* TCP is a reliable transport protocol.
* It uses an acknowledgement mechanism to check the safe and sound arrival of data.

**TCP FEATURES**

* TCP has several features.
* Numbering systems
* Byte number
* Sequence number
* Acknowledgement numbers

**Numbering system:**

* The TCP software keeps track of the segment being transmitted or received, there are two fields is available, called the sequence number and the acknowledgement number,
* These two field refer to a byte number and not a segment number.

**Byte number:**

* The bytes of data being transferred in each connection are numbered by TCP
* The numbering starts with an arbitrarily generated number.

**Sequence number:**

* After the bytes have been numbered, TCP assigns a sequence number to each segment that is being sent.
* The sequence numbers in each direction is defined as follows,

1. The sequence numbers of the first segment is the ISN(initial sequence/number)
2. The sequence number of any other segment is the sequence numbers of the previous segment plus the number of bytes carried by the previous segment.

* Example: suppose a TCP connection is transferring a file of 5000bytes the first bytes is numbered 10001.what are the sequence numbers for each segment if data are sent in five segments. Each carrying 1000 bytes?

**Solution:**

Segments 1 -> sequence number :10001 range: 10001 to 11000

Segments 2 -> sequence number :11001 range: 11001 to 12000

Segments 3 -> sequence number :12001 range: 12001 to 13000

Segments 4 -> sequence number :13001 range: 13001 to 14000

Segments 5 -> sequence number :14001 range: 14001 to 15000

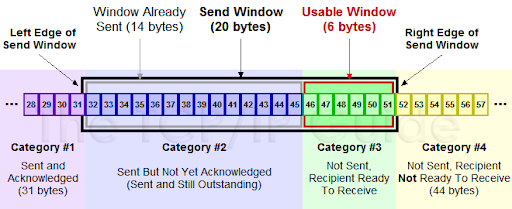
**Acknowledgment numbers:**

* Communication in TCP is full duplex when a connection is established both partices can send and receive data at the send time.
* Each party also uses an acknowledgement numbers to conform the bytes it has received
* The acknowledgement numbers defines the number of the next byte that the party expects to receive.
* In addition the acknowledgement number is cumulative. Which means that the party takes the number of the last byte that it has received safe and sound adds 1 to it

**WINDOWS IN TCP**

* TCP uses two windows,
* Send window
* Receive window
* Four windows for a bidirectional communication
* Two windows for a unidirectional communication.

**Send window:**

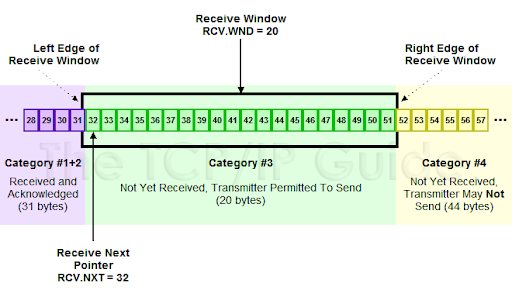


* The window size is 100 bytes.
* The send windows in TCP is similar to the one used with the selective repeat protocol, but with some difference.

1. The window size in SR is the number of packets, but the window size in TCP is the number of bytes.

2. The selective Repeat protocol may use several timer for each packet sent. But TCP protocol uses only one timer.

**Receive window:**

****

* The window Size is 100 bytes.
* There are two difference between the receive window in TCP and the one we used for selective repeat protocol.
* The first difference is that TCP allows the receiving process to pull data at its own place.
* The second difference is the way acknowledgement are used in the TCP protocol TCP uses cumulative acknowledgement mechanism.

**ERROR CONTROL**

* TCP is a reliable transport layer protocol, this means that an application program that delivers a stream of data to TCP relies on TCP to deliver the entire stream to the application program on the other end in order without error and without any part lost or duplicated.
* TCP provides reliability using error control, error control includes mechanisms for detecting and resending corrupted segments, resending lost segments, detecting and discarding duplicated segments.
* Error control in TCP is achieved through the use of following simple tools,
* Checksum
* Acknowledgement

**Checksum:**

* Each segment includes a checksum field which is used to check for a corrupted segment.
* If a segment is corrupted, as detected by an invalid checksum, the segment is discarded by the destination TCP and is considered as lost.

**Acknowledgement:**

* TCP uses acknowledgements to conform the receipt of data segments, control segments that carry no data, but consumes a sequence number are also acknowledged,
* Acknowledge segments are never acknowledged.

**Acknowledgement type:**

* Cumulative acknowledgments(ACK)
* Selective acknowledgement(SACK)

**Cumulative acknowledgement:**

* TCP was originally designed to acknowledge receipt of segments cumulatively.
* The receiver advertise the next byte it expects to receive, ignoring all segments received and stored out of order.

**Selective acknowledgement:**

* More and more implementations are adding another type of acknowledgment called selective acknowledgement or SACK.
* An SACK reports additional information to the sender.
* An SACK reports a block of bytes that is out of order, and also a block of bytes that is duplicated.

**TCP TIMERS**

* To perform their operations smoothly, most TCP implementations use at least four timers.
* Retransmission
* Persistence
* Keep alive
* TIME –WAIT.

**Retransmission timers:**

* To retransmit lost segments TCP employs one retransmission timer, that handle the retransmission time out (RTO) the waiting time for an acknowledgement of a segment we can define the following rules for the retransmission timer.

1. When TCP sends the segments in front of the sending queue, it starts the timer.
2. When the timer expires TCP resend the first segment in front of the queue and restarts the timers
3. When a segment or segments are cumulatively acknowledgment the segment or segments are purged from the queue.
4. If the queue is empty TCP stops the timer, otherwise TCP restarts the timer.

**Persistence timers:**

* To deal with a zero window size advertisement TCP need another timer.
* If the receiving TCP announces a window size of zero, the sending TCP stops transmitting segments until the receiving TCP sends an ACK segment announcing a non zero window size.
* This ACK segment can be lost.
* If this acknowledgment is lost, the receiving TCP thinks that it has done its jobs and wait for the sending TCP to send more segments.

**Keep alive timer:**

* A keep alive timer is used in some implementation to prevent a long idle connections between two TCP’s
* Suppose that a client opens a TCP connection to a server, transfer some data and becomes silent perhaps the client has crashed in this case, the connection remains open forever,
* To remedy this situation most implementations equip a server with a keep alive timer.
* Each time the server hears from a client, it resets the timer the timeout , the timeout is usually 2 hours
* If the server does not hear from the client after 2 hours , if sends a probe segments
* If there is no response after, 10 probes each of which is 75 seconds apart it assumes that the client is down and terminate the connection.

**TIME WAIT timer:**

* The TIME WAIT timer (MSL) is used during connection termination.
* The Maximum Segments Lifetime (MSL) is the amount of time any segment can exist in a network before being discarded.