TRANSPORT LAYER

- This Layer breaks the information data, supplied by Application layer in to smaller units called segments. It numbers every byte in the segment and maintains their accounting.
- This layer ensures that data must be received in the same sequence in which it was sent.
- This layer provides end-to-end delivery of data between hosts which may or may not belong to the same subnet.
- All server processes intend to communicate over the network are equipped with well-known Transport Service Access Points (TSAPs) also known as port numbers.

Functions:

1. **Service Point Addressing:** Transport Layer header includes service point address which is port address. This layer gets the message to the correct process on the computer unlike Network Layer, which gets each packet to the correct computer.
2. **Segmentation and Reassembling:** A message is divided into segments; each segment contains sequence number, which enables this layer in reassembling the message. Message is reassembled correctly upon arrival at the destination and replaces packets which were lost in transmission.
3. **Connection Control:** It includes 2 types:
    - Connectionless Transport Layer : Each segment is considered as an independent packet and delivered to the transport layer at the destination machine.
    - Connection Oriented Transport Layer : Before delivering packets, connection is made with transport layer at the destination machine.
4. **Flow Control:** In this layer, flow control is performed end to end.
5. **Error Control:** Error Control is performed end to end in this layer to ensure that the complete message arrives at the receiving transport layer without any error. Error Correction is done through retransmission.

Design Issues:

- Accepting data from Session layer, split it into segments and send to the network layer.
- Ensure correct delivery of data with efficiency.
- Isolate upper layers from the technological changes.
- Error control and flow control.

The two main Transport layer protocols are:

- **Transmission Control Protocol**

  It provides reliable communication between two hosts.
- **User Datagram Protocol**

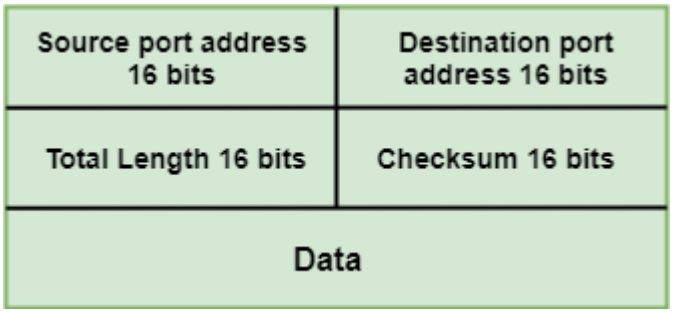
  It provides unreliable communication between two hosts.

UDP

- UDP stands for **User Datagram Protocol**.
- UDP is a simple protocol and it provides nonsequenced transport functionality.
- UDP is a connectionless protocol.
- This type of protocol is used when reliability and security are less important than speed and size.
- UDP is an end-to-end transport level protocol that adds transport-level addresses, checksum error control, and length information to the data from the upper layer.
- The packet produced by the UDP protocol is known as a user datagram.

User Datagram Format

The user datagram has a 16-byte header which is shown below:



Where,

- **Source port address:** It defines the address of the application process that has delivered a message. The source port address is of 16 bits address.
- **Destination port address:** It defines the address of the application process that will receive the message. The destination port address is of a 16-bit address.
- **Total length:** It defines the total length of the user datagram in bytes. It is a 16-bit field.
- **Checksum:** The checksum is a 16-bit field which is used in error detection.

Disadvantages of UDP protocol

- UDP provides basic functions needed for the end-to-end delivery of a transmission.
- It does not provide any sequencing or reordering functions and does not specify the damaged packet when reporting an error.
- UDP can discover that an error has occurred, but it does not specify which packet has been lost as it does not contain an ID or sequencing number of a particular data segment.

TCP

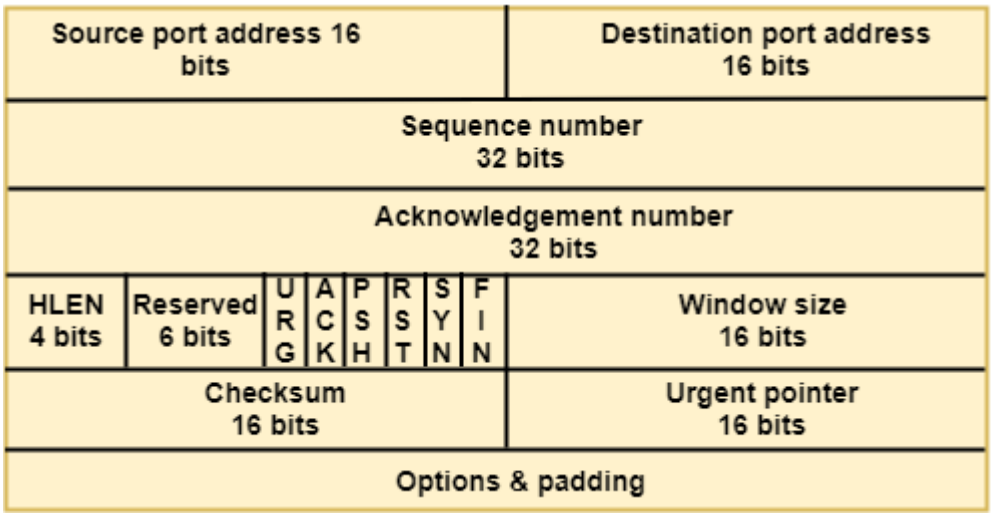
- TCP stands for Transmission Control Protocol.
- It provides full transport layer services to applications.
- It is a connection-oriented protocol means the connection established between both the ends of the transmission. For creating the connection, TCP generates a virtual circuit between sender and receiver for the duration of a transmission.

Features of TCP protocol

- **Stream data transfer:** TCP protocol transfers the data in the form of contiguous stream of bytes. TCP group the bytes in the form of TCP segments and then passed it to the IP layer for transmission to the destination. TCP itself segments the data and forward to the IP.
- **Reliability:** TCP assigns a sequence number to each byte transmitted and expects a positive acknowledgement from the receiving TCP. If ACK is not received within a timeout interval, then the data is retransmitted to the destination.
  The receiving TCP uses the sequence number to reassemble the segments if they arrive out of order or to eliminate the duplicate segments.

- **Flow Control:** When receiving TCP sends an acknowledgement back to the sender indicating the number the bytes it can receive without overflowing its internal buffer. The number of bytes is sent in ACK in the form of the highest sequence number that it can receive without any problem. This mechanism is also referred to as a window mechanism.
- **Multiplexing:** Multiplexing is a process of accepting the data from different applications and forwarding to the different applications on different computers. At the receiving end, the data is forwarded to the correct application. This process is known as demultiplexing. TCP transmits the packet to the correct application by using the logical channels known as ports.
- **Logical Connections:** The combination of sockets, sequence numbers, and window sizes, is called a logical connection. Each connection is identified by the pair of sockets used by sending and receiving processes.
- **Full Duplex:** TCP provides Full Duplex service, i.e., the data flow in both the directions at the same time. To achieve Full Duplex service, each TCP should have sending and receiving buffers so that the segments can flow in both the directions. TCP is a connection-oriented protocol. Suppose the process A wants to send and receive the data from process B. The following steps occur:
    - Establish a connection between two TCPs.
    - Data is exchanged in both the directions.
    - The Connection is terminated.

TCP Segment Format



Where,

- **Source port address:** It is used to define the address of the application program in a source computer. It is a 16-bit field.

- **Destination port address:** It is used to define the address of the application program in a destination computer. It is a 16-bit field.
- **Sequence number:** A stream of data is divided into two or more TCP segments. The 32-bit sequence number field represents the position of the data in an original data stream.
- **Acknowledgement number:** A 32-field acknowledgement number acknowledge the data from other communicating devices. If ACK field is set to 1, then it specifies the sequence number that the receiver is expecting to receive.
- **Header Length (HLEN):** It specifies the size of the TCP header in 32-bit words. The minimum size of the header is 5 words, and the maximum size of the header is 15 words. Therefore, the maximum size of the TCP header is 60 bytes, and the minimum size of the TCP header is 20 bytes.
- **Reserved:** It is a six-bit field which is reserved for future use.
- **Control bits:** Each bit of a control field functions individually and independently. A control bit defines the use of a segment or serves as a validity check for other fields.

There are total six types of flags in control field:

- **URG:** The URG field indicates that the data in a segment is urgent.
- **ACK:** When ACK field is set, then it validates the acknowledgement number.
- **PSH:** The PSH field is used to inform the sender that higher throughput is needed so if possible, data must be pushed with higher throughput.
- **RST:** The reset bit is used to reset the TCP connection when there is any confusion occurs in the sequence numbers.
- **SYN:** The SYN field is used to synchronize the sequence numbers in three types of segments: connection request, connection confirmation (with the ACK bit set), and confirmation acknowledgement.
- **FIN:** The FIN field is used to inform the receiving TCP module that the sender has finished sending data. It is used in connection termination in three types of segments: termination request, termination confirmation, and acknowledgement of termination confirmation.
    - **Window Size:** The window is a 16-bit field that defines the size of the window.
    - **Checksum:** The checksum is a 16-bit field used in error detection.
    - **Urgent pointer:** If URG flag is set to 1, then this 16-bit field is an offset from the sequence number indicating that it is a last urgent data byte.
    - **Options and padding:** It defines the optional fields that convey the additional information to the receiver.

Differences b/w TCP & UDP

Basis for Comparison	TCP	UDP
Definition	TCP establishes a virtual circuit before transmitting the data.	UDP transmits the data directly to the destination computer without verifying whether the receiver is ready to receive or not.
Connection Type	It is a Connection-Oriented protocol	It is a Connectionless protocol
Speed	slow	high
Reliability	It is a reliable protocol.	It is an unreliable protocol.
Header size	20 bytes	8 bytes
acknowledgement	It waits for the acknowledgement of data and has the ability to resend the lost packets.	It neither takes the acknowledgement, nor it retransmits the damaged frame.

SESSION LAYER

The Session Layer allows users on different machines to establish active communication sessions between them.

It's main aim is to establish, maintain and synchronize the interaction between communicating systems. Session layer manages and synchronize the conversation between two different applications. In Session layer, streams of data are marked and are resynchronized properly, so that the ends of the messages are not cut prematurely and data loss is avoided.

Functions:

1. **Dialog Control :** This layer allows two systems to start communication with each other in half-duplex or full-duplex.

2. **Token Management:** This layer prevents two parties from attempting the same critical operation at the same time.
3. **Synchronization :** This layer allows a process to add checkpoints which are considered as synchronization points into stream of data. Example: If a system is sending a file of 800 pages, adding checkpoints after every 50 pages is recommended. This ensures that 50 page unit is successfully received and acknowledged. This is beneficial at the time of crash as if a crash happens at page number 110; there is no need to retransmit 1 to100 pages.

Design Issues:

- To allow machines to establish sessions between them in a seamless fashion.
- Provide enhanced services to the user.
- To manage dialog control.
- To provide services such as **Token management** and **Synchronization**.

Remote Procedure Call

Remote Procedure Call (RPC) provides a different paradigm for accessing network services. Instead of accessing remote services by sending and receiving messages, a client invokes services by making a local procedure call. The local procedure hides the details of the network communication.

When making a remote procedure call:

1. The calling environment is suspended, procedure parameters are transferred across the network to the environment where the procedure is to execute, and the procedure is executed there.
2. When the procedure finishes and produces its results, its results are transferred back to the calling environment, where execution resumes as if returning from a regular procedure call.

The main goal of RPC is to hide the existence of the network from a program. As a result, RPC doesn't quite fit into the OSI model:

1. The message-passing nature of network communication is hidden from the user. The user doesn't first open a connection, read and write data, and then close the connection. Indeed, a client often doesn not even know they are using the network!
2. RPC often omits many of the protocol layers to improve performance. Even a small performance improvement is important because a program may invoke RPCs often. For example, on (diskless) Sun workstations, every file access is made via an RPC.

RPC is especially well suited for client-server (e.g., query-response) interaction in which the flow of control alternates between the caller and callee. Conceptually, the client and server do not both execute at the same time. Instead, the thread of execution jumps from the caller to the callee and then back again.

The following steps take place during an RPC:

1. A client invokes a *client stub* procedure, passing parameters in the usual way. The client stub resides within the client's own address space.
2. The client stub *marshalls* the parameters into a message. Marshalling includes converting the representation of the parameters into a standard format, and copying each parameter into the message.
3. The client stub passes the message to the transport layer, which sends it to the remote server machine.
4. On the server, the transport layer passes the message to a *server stub*, which demarshalls the parameters and calls the desired server routine using the regular procedure call mechanism.
5. When the server procedure completes, it returns to the server stub (e.g., via a normal procedure call return), which marshalls the return values into a message. The server stub then hands the message to the transport layer.
6. The transport layer sends the result message back to the client transport layer, which hands the message back to the client stub.
7. The client stub demarshalls the return parameters and execution returns to the caller.

PRESENTATION LAYER

The primary goal of this layer is to take care of the **syntax** and **semantics** of the information exchanged between two communicating systems.

Presentation layer takes care that the data is sent in such a way that the receiver will understand the information(data) and will be able to use the data. Languages(syntax) can be different of the two communicating systems. Under this condition presentation layer plays a role translator.

Functions:

1) Translation

- The processes or running programs in two machines are usually exchanging the information in the form of numbers, character strings and so on before being transmitted. The information should be changed to bitstreams because different computers use different encoding schemes.
- The Presentation layer is responsible for compatibility between these encoding methods.
- The Presentation layer at the sender's side changes the information from its sender dependent format.
- The Presentation layer at the receiving machine changes the common format into its receivers dependent format.

Example: Convert ASCII code to EBCDIC code.

2) Encryption

- The system must be able to assure privacy regarding the message or information as it also carries sensitive information.
- Encryption means that the sender transforms the original information or message to another form, this data after encryption is known as the ciphertext and this ciphertext sends the resulting message out over the network.
- Decryption concerned with the transform of the message back to its original form. This decrypted data is known as plain text.

3) Compression

- Data Compression means reduces the number of bits to be transmitted by this reduce the bandwidth of the data.
- Data Compression becomes particularly important in the transmission of multimedia such as audio, video, text, etc.

Design Issues:

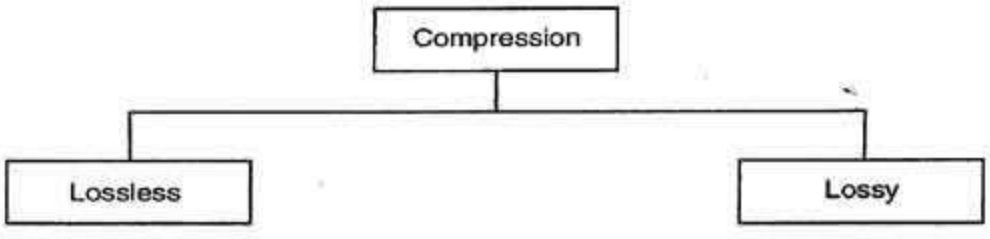
- To manage and maintain the **Syntax** and **Semantics** of the information transmitted.
- **Encoding data** in a standard agreed upon way. Eg: String, double, date, etc.
- Perform **Standard Encoding** on wire.

Data compression

Data compression is the function of presentation layer in OSI reference model. Compression is often used to maximize the use of bandwidth across a network or to optimize disk space when saving data.

There are two general types of compression algorithms:

1. Lossless compression
2. Lossy compression



Types of Compression

Lossless Compression

- Lossless compression compresses the data in such a way that when data is decompressed it is exactly the same as it was before compression *i.e.* there is no loss of data.
- A lossless compression is used to compress file data such as executable code, text files, and numeric data, because programs that process such file data cannot tolerate mistakes in the data.
- Lossless compression will typically not compress file as much as lossy compression techniques and may take more processing power to accomplish the compression.

Lossless Compression Algorithms

- The various algorithms used to implement lossless data compression are :

1. Run length encoding
2. Differential pulse code modulation
3. Dictionary based encoding

1. Run length encoding

• This method replaces the consecutive occurrences of a given symbol with only one copy of the symbol along with a count of how many times that symbol occurs. Hence the names ‘run length'.

• For example, the string AAABBCDDDD would be encoded as 3A2BIC4D.

• A real life example where run-length encoding is quite effective is the fax machine. Most faxes are white sheets with the occasional black text. So, a run-length encoding scheme can take each line and transmit a code for while then the number of pixels, then the code for black and the number of pixels and so on.

• This method of compression must be used carefully. If there is not a lot of repetition in the data then it is possible the run length encoding scheme would actually increase the size of a file.

2. Differential pulse code modulation

• In this method first a reference symbol is placed. Then for each symbol in the data, we place the difference between that symbol and the reference symbol used.

• For example, using symbol A as reference symbol, the string AAABBC DDDD would be encoded as AOOOl123333, since A is the same as reference symbol, B has a difference of 1 from the reference symbol and so on.

3. Dictionary based encoding

• One of the best known dictionary based encoding algorithms is Lempel-Ziv (LZ) compression algorithm.

• This method is also known as substitution coder.

• In this method, a dictionary (table) of variable length strings (common phrases) is built.

• This dictionary contains almost every string that is expected to occur in data.

• When any of these strings occur in the data, then they are replaced with the corresponding index to the dictionary.

• In this method, instead of working with individual characters in text data, we treat each word as a string and output the index in the dictionary for that word.

• For example, let us say that the word "compression" has the index 4978 in one particular dictionary; it is the 4978^{th} word is usr/share/dict/words. To compress a body of text, each time the string "compression" appears, it would be replaced by 4978.

Lossy Compression

- Lossy compression is the one that does not promise that the data received is exactly the same as data send *i.e.* the data may be lost.

- This is because a lossy algorithm removes information that it cannot later restore.
- Lossy algorithms are used to compress still images, video and audio.
- Lossy algorithms typically achieve much better compression ratios than the lossless algorithms.

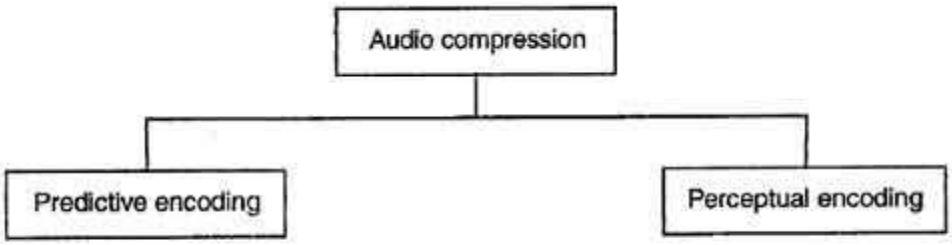
Audio Compression

• Audio compression is used for speech or music.

• For speech, we need to compress a 64-KHz digitized signal; For music, we need to compress a 1.411.MHz signal

Two types of techniques are used for audio compression:

1. Predictive encoding
2. Perceptual encoding



Techniques of audio compression

Predictive encoding

• In predictive encoding, the differences between the samples are encoded instead of encoding all the sampled values.

• This type of compression is normally used for speech.

• Several standards have been defined such as GSM (13 kbps), G. 729 (8 kbps), and G.723.3 (6.4 or 5.3 kbps).

Perceptual encoding

• Perceptual encoding scheme is used to create a CD-quality audio that requires a transmission bandwidth of 1.411 Mbps.

• MP3 (MPEG audio layer 3), a part of MPEG standard uses this perceptual encoding.

• Perceptual encoding is based on the science of psychoacoustics, a study of how people perceive sound.

• The perceptual encoding exploits certain flaws in the human auditory system to encode a signal in such a way that it sounds the same to a human listener, even if it looks quite different on an oscilloscope.

• The key property of perceptual coding is that some sounds can mask other sound. For example, imagine that you are broadcasting a live flute concert and all of a sudden someone starts striking a hammer on a metal sheet. You will not be able to hear the flute any more. Its sound has been masked by the hammer.

• Such a technique explained above is called frequency masking-the ability of a loud sound in one frequency band to hide a softer sound in another frequency band that would have been audible in the absence of the loud sound.

• Masking can also be done on the basis of time. For example: Even if the hammer is not striking on a metal sheet, the flute will be inaudible for a short period of time because the ears turn down its gain when they start and take a finite time to turn up again.

• Thus, a loud sound can numb our ears for a short time even after the sound has stopped. This effect is called temporal masking.

MP3

• MP3 uses these two phenomena, *i.e.* frequency masking and temporal masking to compress audio signals.

• In such a system, the technique analyzes and divides the spectrum into several groups. Zero bits are allocated to the frequency ranges that are totally masked.

• A small number of bits are allocated to the frequency ranges that are partially masked.

• A larger number. of bits are allocated to the frequency ranges that are not masked.

• Based on the range of frequencies in the original analog audio, MP3 produces three data rates: 96kbps, 128 kbps and 160 kbps.

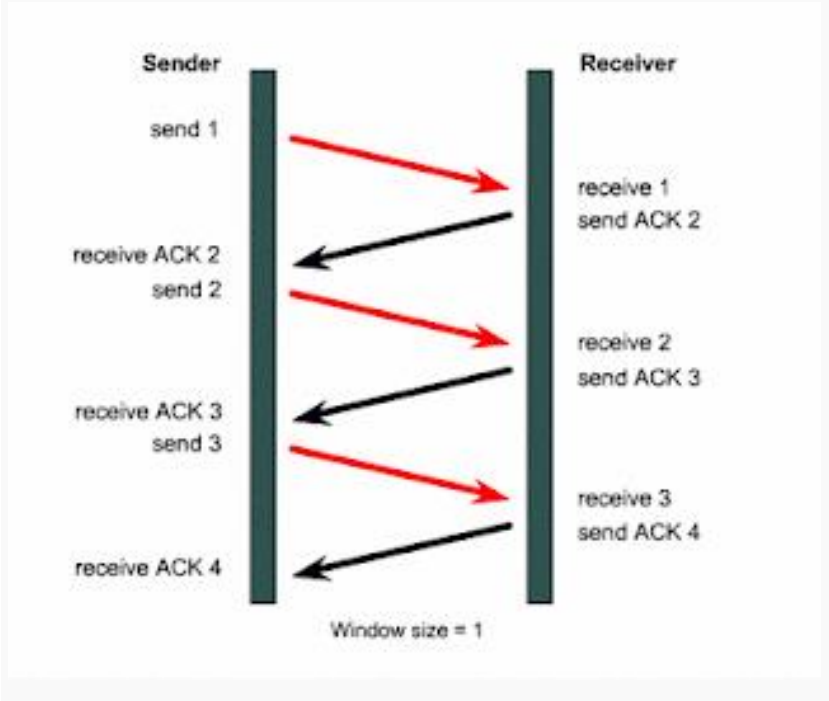
TCP-window management

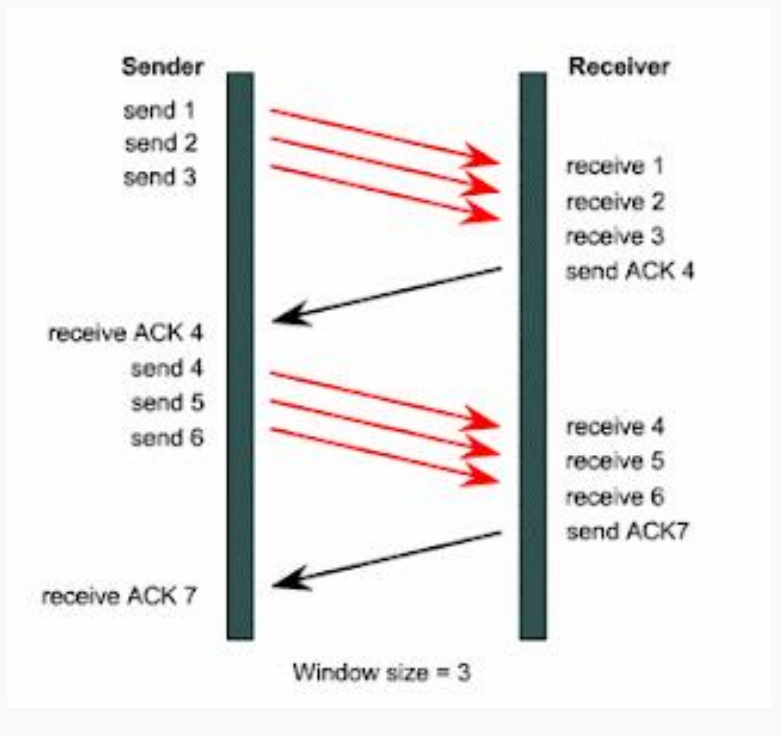
Window management in TCP is an important concept that ensures reliability in packet delivery as well as reduce the wastage of time in waiting for the acknowledge after each packet.

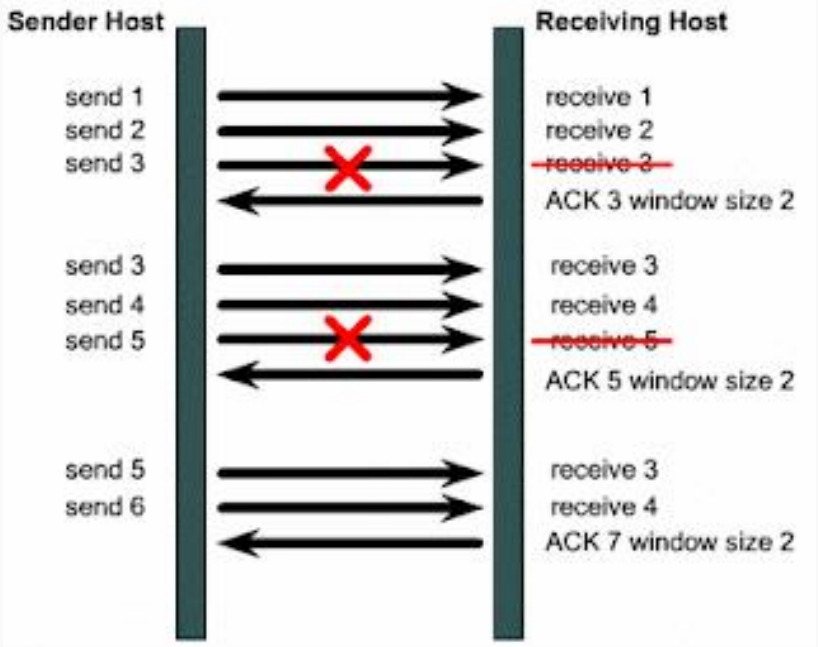
Window size: window size determines the amount of data that you can transmit before receiving an acknowledgment. Sliding window refers to the fact that the window size is negotiated dynamically during the TCP session.

1. Expectational acknowledgment means that the acknowledgment number refers to the octet that is next expected
2. If the source receives no acknowledgment, it knows to retransmit at a slower rate.

The mechanism of the sliding window style may be understood easily with the help of below given diagrams:




Sender
Receiver
send 1
send 2
send 3
receive 1
receive 2
receive 3
send ACK 4
receive ACK 4
send 4
send 5
send 6
receive 4
receive 5
receive 6
send ACK7
receive ACK 7
Window size = 3


Sender Host
Receiving Host
send 1
send 2
send 3
receive 1
receive 2
receive 3
ACK 3 window size 2
send 3
send 4
send 5
receive 3
receive 4
receive 5
ACK 5 window size 2
send 5
send 6
receive 3
receive 4
ACK 7 window size 2