4.3.5 Flow Control

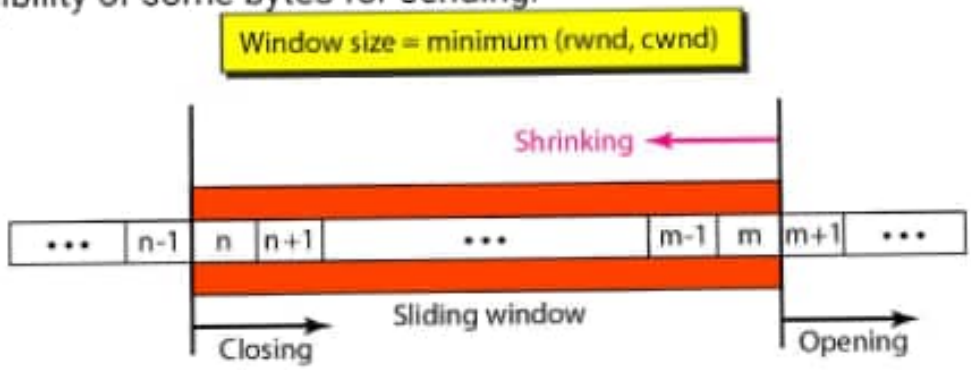
TCP uses a sliding window, to handle flow control. The sliding window protocol used by TCP, however, is something between the *Go-Back-N* and Selective Repeat sliding window. The sliding window protocol in TCP looks like the Go- Back- N protocol because it does not use NAKs; it looks like Selective Repeat because the receiver holds the out-of-order segments until the missing ones arrive.

There are two big differences between this sliding window and the one we used at the data link layer. First, the sliding window of TCP is byte-oriented; the one we discussed in the data link layer is frame-oriented. Second, the TCP's sliding window is of variable size; the one we discussed in the data link layer was of fixed size.

Below figure shows the sliding window in TCP. The window spans a portion of the buffer containing bytes received from the process. The bytes inside the window are the bytes that can be in transit; they can be sent without worrying about acknowledgment. The imaginary window has two walls: one left and one right.

The window is *opened, closed,* or *shrunk.* These three activities, as we will see, are in the control of the receiver (and depend on congestion in the network), not the sender. The sender must obey the commands of the receiver in this matter.

Opening a window means moving the right wall to the right. This allows more new bytes in the buffer that are eligible for sending. Closing the window means moving the left wall to the right. This means that some bytes have been acknowledged and the sender need not worry about them anymore. Shrinking the window means moving the right wall to the left. This is strongly discouraged and not allowed in some implementations because it means revoking the eligibility of some bytes for sending.



Some points about TCP sliding windows:

- The size of the window is the lesser of *rwnd* and *cwnd.*
- The source does not have to send a full window's worth of data.
- The window can be opened or closed by the receiver, but should not be shrunk.
- The destination can send an acknowledgment at any time as long as it does not result in a shrinking window.
- The receiver can temporarily shut down the window; the sender, however, can always send a segment of 1 byte after the window is shut down.

Congestion Control

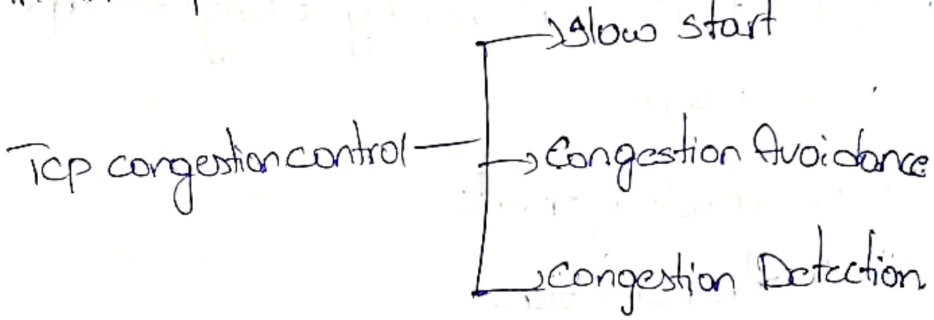
Congestion control is a mechanism that controls the entry of data packets into the network, enabling a better use of a shared network infrastructure.

Reasons for the cause of congestion TCP are

(i) Receiver window size
(ii) Congestion window size.

Congestion control in TCP

TCP's general mechanism for handling congestion are in three phases



Slow start phase

(i) Initially sender sets,
Congestion window size = Maximum Segment Size(MSS)

(ii) After receiving each acknowledgment, sender increases congestion window size by 1 MSS

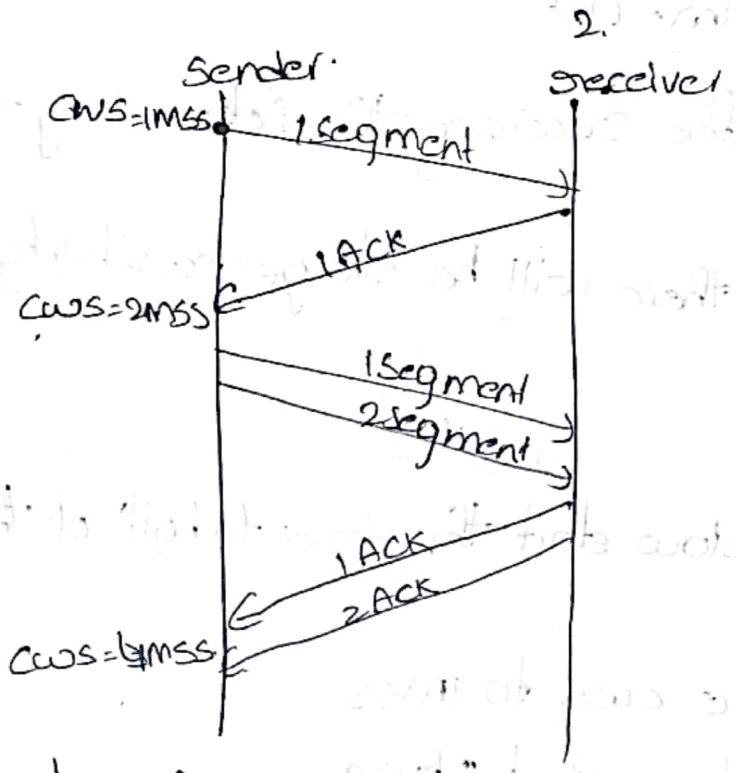
(iii) In this phase congestion window increases exponentially

Congestion window size = CWS + Maximum. (CWS)

From figure: After 1 round trip time, CWS = $(2)^1 = 2$ MSS
After 2 round trip time, CWS = $(2)^2 = 4$ MSS.
and so on.

This phase continues until the cwnd reaches the slow start threshold.

Threshold = (Receiver Window Size / Maximum Segment size)



2) Congestion Avoidance

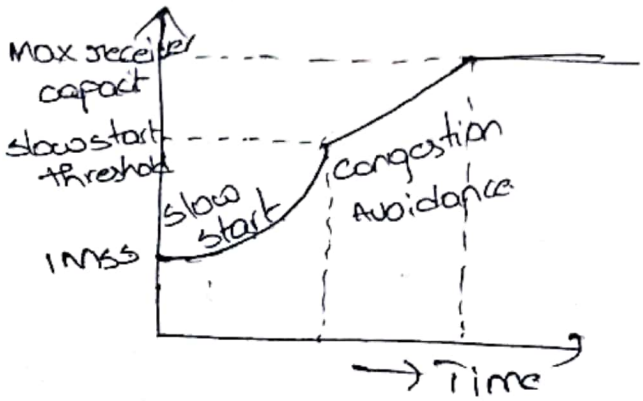
After receiving threshold:

i) Sender increases the cwnd to avoid congestion

ii) On receiving each Acknowledgement, sender increments the cwnd by 1.

$$cwnd = cwnd + 1$$

This continues until cwnd becomes equal to receiver window size.



3) Congestion Detection

This can be done in two different situations

i, Detection on Time Out

Time expires before receiving the Acknowledgement for a segment.

At this case there will be stronger possibility in network.

~~This can be~~

Reaction

i, setting the slow start threshold to half of the current cws.

ii, Decreasing the cws to 1mss

iii, Resuming slow start phase.

ii, Detection on Receiving 3 Duplicate acknowledgement

Sender receives 3 duplicate Acknowledgements

It has less chance for congestion in the network

Rection

i, setting slow start threshold to half of cws

ii, Resuming the congestion avoidance phase.

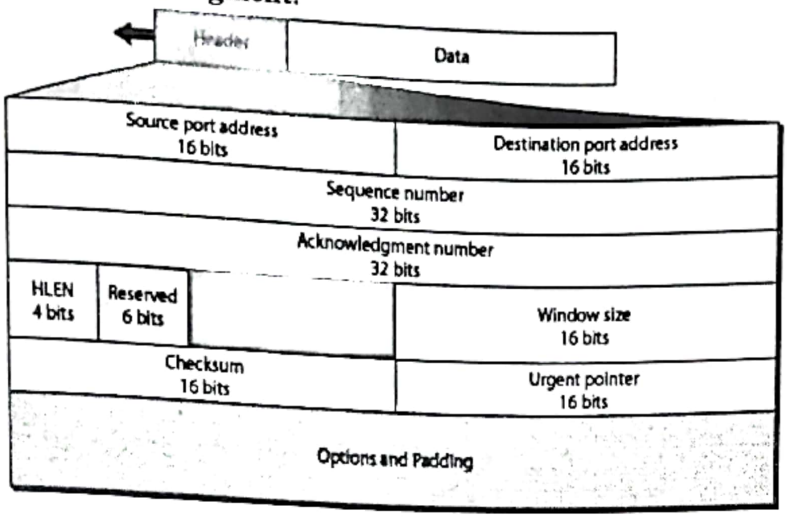
	TCP	UDP
Full form	It stands for **Transmission Control Protocol.**	It stands for **User Datagram Protocol.**
Type of connection	It is a connection-oriented protocol, which means that the connection needs to be established before the data is transmitted over the network.	It is a connectionless protocol, which means that it sends the data without checking whether the system is ready to receive or not.
Reliable	TCP is a reliable protocol as it provides assurance for the delivery of data packets.	UDP is an unreliable protocol as it does not take the guarantee for the delivery of packets.

Speed	TCP is slower than UDP as it performs error checking, flow control, and provides assurance for the delivery of	UDP is faster than TCP as it does not guarantee the delivery of data packets.
Header size	The size of TCP is 20 bytes.	The size of the UDP is 8 bytes.
Acknowledgment	TCP uses the three-way-handshake concept. In this concept, if the sender receives the ACK, then the sender will send the data. TCP also has the ability to resend the lost data.	UDP does not wait for any acknowledgment; it just sends the data.

	data.	
Flow control mechanism	It follows the flow control mechanism in which too many packets cannot be sent to the receiver at the same time.	This protocol follows no such mechanism.
Error checking	TCP performs error checking by using a checksum. When the data is corrected, then the data is retransmitted to the receiver.	It does not perform any error checking, and also does not resend the lost data packets.
Applications	This protocol is mainly used where a secure and reliable communication process is required, like military services, web browsing, and e-mail.	This protocol is used where fast communication is required and does not care about the reliability like VoIP, game streaming, video and music streaming, etc.

4.3.3 Segment

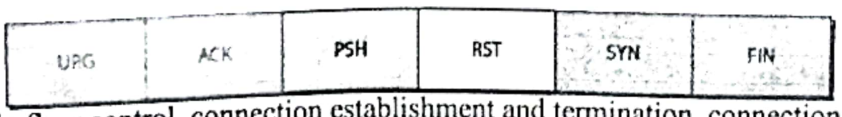
A packet in TCP is called a **segment.**



The segment consists of a 20- to 60-byte header, followed by data from the application program. The header is 20 bytes if there are no options and up to 60 bytes if it contains options. We will discuss some of the header fields in this section.

- **Source port address**. This is a 16-bit field that defines the port number of the application program in the host that is sending the segment. This serves the same purpose as the source port address in the UDP header.
- **Destination port address.** This is a 16-bit field that defines the port number of the application program in the host that is receiving the segment. This serves the same purpose as the destination port address in the UDP header.
- **Sequence number**. This 32-bit field defines the number assigned to the first byte of data contained in this segment. As we said before, TCP is a stream transport protocol. To ensure connectivity, each byte to be transmitted is numbered. The sequence number tells the destination which byte in this sequence comprises the first byte in the segment. During connection establishment, each party uses a random number generator to create an initial sequence number (ISN), which is usually different in each direction.
- **Acknowledgment number.** This 32-bit field defines the byte number that the receiver of the segment is expecting to receive from the other party. If the receiver of the segment has successfully received byte number x from the other party, it defines x + I as the acknowledgment number. Acknowledgment and data can be piggybacked together.
- **Header length**. This 4-bit field indicates the number of 4-byte words in the TCP header. The length of the header can be between 20 and 60 bytes. Therefore, the value of this field can be between 5 (5 x 4 =20) and 15 (15 x 4 =60).
- **Reserved**. This is a 6-bit field reserved for future use.
- **Control**. This field defines 6 different control bits or flags as shown in below figure. One or more of these bits can be set at a time.

URG: Urgent pointer is valid
ACK: Acknowledgment is valid
PSH: Request for push
RST: Reset the connection
SYN: Synchronize sequence numbers
FIN: Terminate the connection



These bits enable flow control, connection establishment and termination, connection abortion, and the mode of data transfer in TCP. A brief description of each bit is shown in below table.

Flag	*Description*
URG	The value of the urgent pointer field is valid.
ACK	The value of the acknowledgment field is valid.
PSH	Push the data.
RST	Reset the connection.
SYN	Synchronize sequence numbers during connection.
FIN	Terminate the connection.

- **Window size.** This field defines the size of the window, in bytes, that the other party must maintain. Note that the length of this field is 16 bits, which means that the maximum size of the window is 65,535 bytes. This value is normally referred to as the receiving window (rwnd) and is determined by the receiver. The sender must obey the dictation of the receiver in this case.
- **Checksum**. This 16-bit field contains the checksum. The calculation of the checksum for TCP follows the same procedure as the one described for UDP. However, the inclusion of the checksum in the UDP datagram is optional, whereas the inclusion of the checksum for TCP is mandatory. The same pseudoheader, serving the same purpose, is added to the segment. For the TCP pseudoheader, the value for the protocol field is 6.
- **Urgent pointer**. This 16-bit field, which is valid only if the urgent flag is set, is used when the segment contains urgent data. It defines the number that must be added to the sequence number to obtain the number of the last urgent byte in the data section of the segment.
- **Options**. There can be up to 40 bytes of optional information in the TCP header.