**Connection oriented protocol or connection-less protocol**

TCP is a connection oriented protocol. Difference between a connection-oriented protocol and a connection-less protocol is that a connection-oriented protocol does not send any data until a proper connection is established. Connection establishment refers to the process of initializing protocol specific features.

TCP, in connection establishment process, initializes sequence and acknowledgement numbers. TCP refers this process as the three-way handshake process.

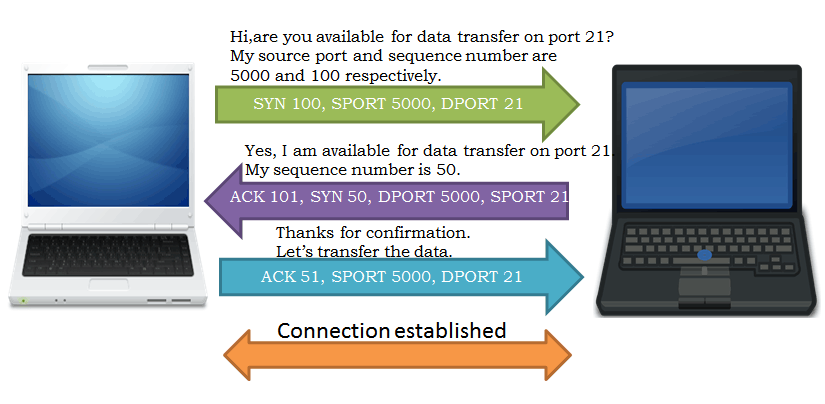
**TCP Three-way handshake process**

Source sends a SYN (synchronization) segment to the destination. This segment indicates that source want to establish a reliable session with destination.

Destination responds back with a SYN/ACK (synchronization/acknowledgement) segment. This segment indicates that destination received the source’s connection request and ready to setup a reliable session with source.

Upon receiving a SYN/ACK segment from destination, source sends an ACK (acknowledgement) segment to the destination. This segment indicates that source received the confirmation from destination and the session is now fully reliable.

Following figure shows an example of Three-way handshake process.



Once the three-way handshake process is done both source and destination can transfer the data.

TCP does not send any data without establishing proper connection. Segments which are used in connection establishment or three-way handshake process contain only the header information that is used to initialize the TCP specific features. These features are explained below.

**Reliability through acknowledgement and sequencing**

The term TCP reliability covers mainly following items;

* Recognizing and resending lost packets
* Detecting and dropping duplicate packets
* Recognizing and reordering the packets that arrive out of order
* Controlling the overflow of segments

To provide reliability, TCP assigns a sequence number to each sent segment. This number not only helps the destination host in reordering any incoming segments that arrived out of the order but also help in verifying that all sent segments were received.

Acknowledgement numbers are used in opposite direction. These numbers are used to send the verification of received segments, notification of lost segments and acknowledgement for next segments.

Upon receiving all sent segments, to get the next segments, destination sends a segment with a number in the acknowledgment field that is one number higher than the received sequence number.

If any segment lost in transmission, its sequence number is used in acknowledgment field to notify the sender about it.

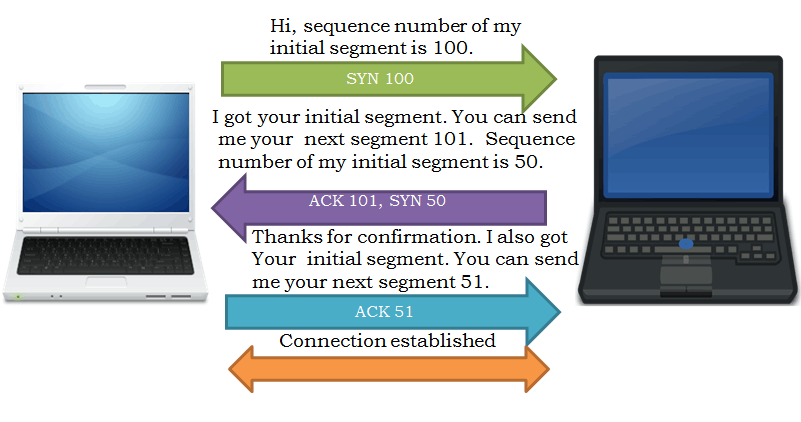
Both, sequence number and acknowledge number are initialized in three-way handshake process. Source and destination hosts update each other about their sequence number in this process.

Once source and host know each other’s sequence numbers, they use them in data exchange process. Before we take an example of this process, let’s understand one more number that is also initialized in three-way handshake process and is used with these numbers.

**Flow control through Windowing**

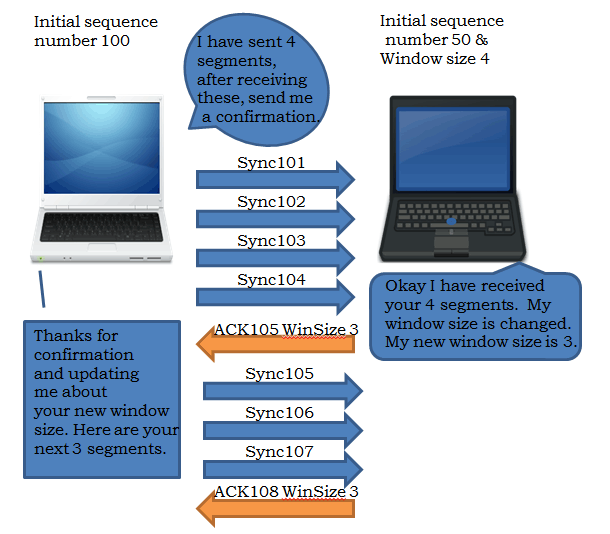
Windowing is the process of controlling the flow of segments. It ensures that one host doesn’t flood another host with too many segments, overflowing its receiving buffer.

In three-way handshake process, receiver computer, while responding to the initial segment, updates the sender computer about its window size. Window size is the number of segments that it can store in its buffer. Once sender computer knows the window size of receiver computer, it can control the flow of segments.



For example, if window size of the receiver computer is 4, sender computer sends only the 4 segments. Once 4 segments are sent, it waits for confirmation from receiver computer before sending next 4 segments.

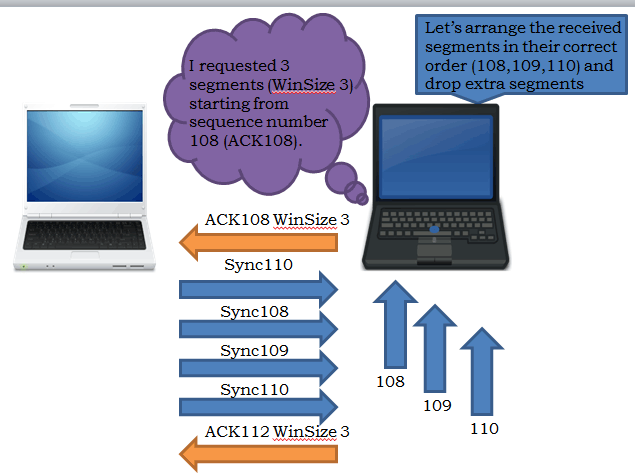
While sending confirmation, receiver computer can change the window size. For example, it can ask sender computer to send more or less segments in next section. This feature is called sliding windowing or dynamic windowing. It allows receiver to control the flow of segments that sender computer can send.



**Reordering segments and in correct order and dropping extra segments**

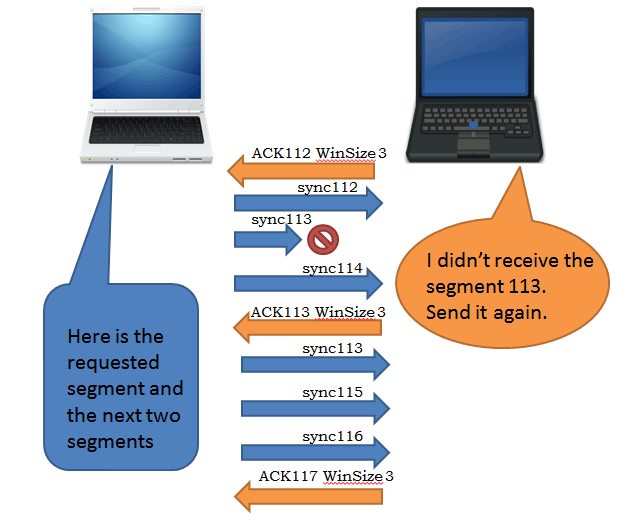
To arrange the arrived segments in correct order, receiver computer uses the sequence numbers of the segments. To detect and drop the duplicate or extra segments, it compares the received segments with the requested segments. For example if receiver computer requested 3 segments by specifying window size 3 in acknowledgment and received 4 segments, it assumes that one segment is arrived extra.

Sequence numbers also help in detecting duplicate segments. If two or more segments have same sequence numbers, they are duplicates. Duplicate segments are dropped.



**Recovering lost segments**

As mentioned above, receiver computer compares the arrived segments with the expected segments. If it finds any segment is missing, it uses the sequence number of that segment to acknowledge the receiver computer about it. When sender computer receives an acknowledgment of a segment that it has already sent, it assumes that acknowledged segment has lost. While transmitting the next set of segments (segments equal to the window size), it retransmits the lost segments first and new segments later. For example if window size is 3 and lost segment is 1, then the transmitted segments will be; lost segment, new segment and new segment.



**Comparing with UDP**

UDP is a connection-less protocol. It does not establish any connection or exchange any messages before sending the data.

TCP protocol always starts a session with the three-way handshake process. It means if an application wants to send its data through TCP, it has to wait until the proper connection is established through the three-way hand shake process.

UDP doesn’t use any mechanism or process before starting the session. It means if an application wants to send its data through UDP, it can send its data immediately without any delay.

UDP neither sequences the segments nor care about the order in which they are sent to the destination. It also does not take acknowledgement of the sent segments to verify that they reached at destination. It just sends and forgets about them. Because of this, it is also referred as an unreliable protocol.

UDP takes less bandwidth and uses fewer processing cycles in comparison of TCP.