OSI Reference Model Transport Layer

TCP Sequence Number & Wrap Around Time

Munesh Singh

Indian Institute of Information Technology, Design and Manufacturing Kancheepuram,
Chennai, Tamil Nadu 600127

November 16, 2020

TCP Sequence Number

- TCP continuously receives data from the application layer.
- It divides the data into chunks where each chunk is a collection of bytes.
- It then creates TCP segments by adding a TCP header to the data chunks.

  TCP segment = TCP header + Data chunk
- Each TCP segment sent by the sender contains some bytes of data.
- TCP assigns a unique number to each data byte for its identification.
- This unique number is called as TCP Sequence Number.
- **Purpose**
  - It helps to identify each data byte uniquely.
  - It helps in the segmentation of data into TCP segments and reassemble them later.
  - It helps to keep track of how much data has been transferred and received.
  - It helps to put the data back into the correct order if
  - it is received in the wrong order.
  - It helps to request data when it has been lost in transit.

Maximum Number of Sequence Numbers

- In TCP header, sequence number is a 32 bit field.
- So, maximum number of possible sequence numbers $= 2^{32}$.
- These sequence numbers lie in the range [0 , 2^{32} 1].
- **Note**
  - Maximum number of possible sequence numbers $= 2^{32}$.
  - This does not imply that only 2^{32} bytes = 4 GB data can be sent using TCP.
  - The concept of wrap around allows to send unlimited data using TCP.
- **Concept Of Wrap Around**
  - After all the 2^{32} sequence numbers are used up and more data is to be sent
  - The sequence numbers can be wrapped around and used again from the starting.
    - If the initial sequence number chosen is X.
    - Then sequence numbers are used from X to 2^{32} 1 and then from to 0 to X-1.
    - Then, sequence numbers are wrapped around to send more data.

- Consider the initial sequence number used is 0.
- Then after sending 4 GB data, all the sequence numbers would get used up.
- To send more data, sequence numbers are reused from the starting.
- Wrapping around can be done again and again to send more and more data.
- **Wrap Around Time**
  - Time taken to use up all the 2^{32} sequence numbers is called as wrap around time.
  - It depends on the bandwidth of the network i.e. the rate at which the bytes go out.
  - More the bandwidth, lesser the wrap around time and vice versa.
    Wrap Around Time $\propto$ 1 / Bandwidth
- **Formula**
  - If bandwidth of the network = x bytes/sec, then-

$$\text{Wrap Around Time} = \frac{2^{32}}{x} \text{ sec}$$

Life Time Of TCP Segment

- In modern Computer
  - Life time of a TCP segment is 180 seconds or 3 minutes.
  - It means after sending a TCP segment, it might reach the receiver taking 3 minutes in the worst case.
- **How Wrap Around Is Possible?**
  - The life time of a TCP segment is just 180 seconds.
  - Wrap around time is much greater than life time of a TCP segment.
  - So, by the time the sequence numbers wrap around, there is no probability of existing any segment having the same sequence number.
  - Thus, even after wrapping around, the sequence number of all the bytes will be unique at any given time.
- **Reducing Wrap Around Time**
  - Wrap around time can be reduced to the life time of a TCP segment.
    - After the life time of a segment completes, it is considered that the segment no longer exists.
    - So, sequence numbers used by the segment frees up and can be reused.
    - There must exist as many sequence numbers as there are number of data bytes sent in time equal to life time of segment.

- **Formula**

  Number of bits required in the sequence number field so that wrap around time becomes equal to lifetime of TCP segment = log2 (lifetime of TCP segment x Bandwidth)

  - The number of bits will be greater than 32 bits.
  - The extra bits are appended in the Options field of TCP header.

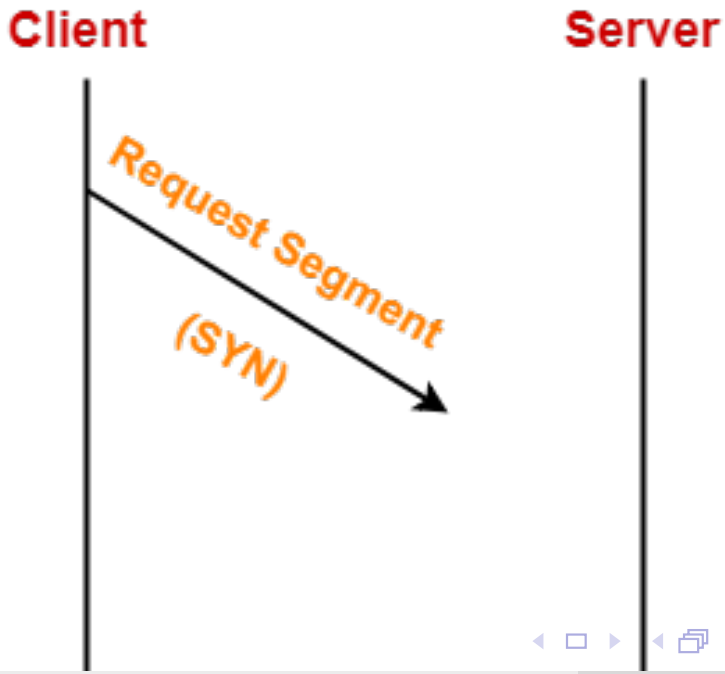
3 Way Handshake

- TCP establishes an end to end connection between the sender and receiver.
- This connection is established before exchanging the data.
- TCP connection is reliable and ensures in order delivery.

3 Way Handshake

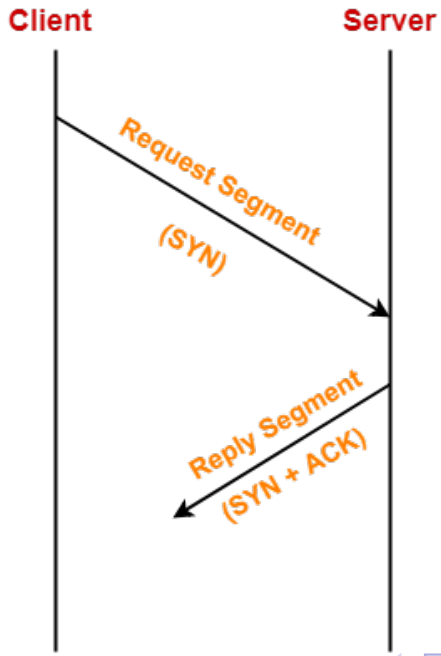
- Three Way Handshake is a process used for establishing a TCP connection.
  - Client wants to establish a connection with the server.
  - Before Three Way Handshake, both client and server are in closed state.

- TCP Handshake involves the following steps in establishing the connection-
  - **Step-01: SYN-**
    - Client sends a request segment to the server.
    - Request segment consists only of TCP Header with an empty payload.
    - Then, it waits for a reply segment from the server.
  - **Request segment contains the following information in TCP header-**
    - Initial sequence number
    - SYN bit set to 1
    - Maximum segment size
    - Receiving window size



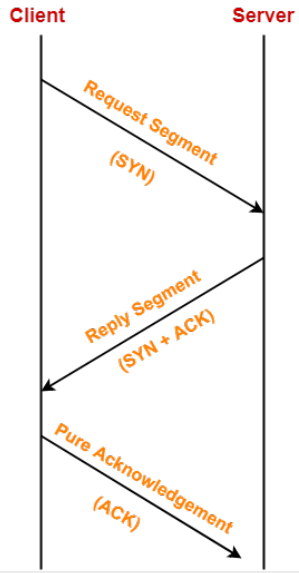
- **Initial Sequence Number**
  - Client sends the initial sequence number to the server.
  - It is contained in the sequence number field.
  - It is a randomly chosen 32 bit value.
- **SYN Bit Set To 1**
  - Client sets SYN bit to 1 which indicates the server-
    - This segment contains the initial sequence number used by the client.
    - It has been sent for synchronizing the sequence numbers.
- **Maximum Segment Size (MSS)**
  - Client sends its MSS to the server.
  - It dictates the size of the largest data chunk that client can send and receive from the server.
  - It is contained in the Options field.
- **Receiving Window Size**
  - Client sends its receiving window size to the server.
  - It dictates the limit of unacknowledged data that can be sent to the client.
  - It is contained in the window size field.

- **Step-02: SYN + ACK**
  - After receiving the request segment,
    - Server responds to the client by sending the reply segment.
    - It informs the client of the parameters at the server side.
  - **Reply segment contains the following information in TCP header-**
    - Initial sequence number
    - SYN bit set to 1
    - Maximum segment size
    - Receiving window size
    - Acknowledgment number
    - ACK bit set to 1



- **Initial Sequence Number-**
  - Server sends the initial sequence number to the client.
  - It is contained in the sequence number field.
  - It is a randomly chosen 32 bit value.
- **SYN Bit Set To 1**
  - Server sets SYN bit to 1 which indicates the client-
    - This segment contains the initial sequence number used by the server.
    - It has been sent for synchronizing the sequence numbers.
- **Maximum Segment Size (MSS)**
  - Server sends its MSS to the client.
  - It dictates the size of the largest data chunk that server can send and receive from the client.
  - It is contained in the Options field.
- **Receiving Window Size**
  - Server sends its receiving window size to the client.
  - It dictates the limit of unacknowledged data that can be sent to the server.
  - It is contained in the window size field.

- **Acknowledgement Number**
  - Server sends the initial sequence number incremented by 1 as an acknowledgement number.
  - It dictates the sequence number of the next data byte that server expects to receive from the client.
- **ACK Bit Set To 1**
  - Server sets ACK bit to 1.
  - It indicates the client that the acknowledgement number field in the current segment is valid.
- **Step-03: ACK**
  - After receiving the reply segment,
    - Client acknowledges the response of server.
    - It acknowledges the server by sending a pure acknowledgement.



- **Important Points**
  - In step-01 and step-02-
    - The connection parameters are established for the first side.
    - They are acknowledged by the second side.
- **In step-02 and step-03**
  - The connection parameters are established for the second side.
  - They are acknowledged by the first side.
- Connection establishment phase consume 1 sequence number of both the sides.
  - Request segment consumes 1 sequence number of the requester.
  - Reply segment consumes 1 sequence number of the respondent.
  - Pure acknowledgments do not consume any sequence number.
- **Pure acknowledgement for the reply segment is not necessary.**
  - If client sends the data packet immediately, then it will be considered as an acknowledgement.
  - It means that in the first two steps only, the full duplex connection is established.

- **For all the segments except the request segment, ACK bit is always set to 1.**
  - For the request segment, acknowledgement number field will always be invalid.
  - For all other segments, acknowledgement number field will always be valid.
- **Certain parameters are negotiated during connection establishment.**
  - Window size
  - Maximum segment size
  - Timer values
- In any TCP segment
  - If SYN bit = 1 and ACK bit = 0, then it must be the request segment.
  - If SYN bit = 1 and ACK bit = 1, then it must be the reply segment.
  - If SYN bit = 0 and ACK bit = 1, then it can be the pure ACK or segment meant for data transfer.
  - If SYN bit = 0 and ACK bit = 0, then this combination is not possible.

- The combination SYN bit = 0 and ACK bit = 0 is not possible.
- It is because SYN bit = 0 signifies it is not the request segment and reply segment.
- For all other segments, ACK bit is always set to 1.
- Consider sender sends the segments of size greater than MSS of receiver.
- Then, they are first fragmented first at the receiver side.
- It causes an extra overhead.
- There is no dedicated field for sending MSS in TCP header.
- This is because MSS has to informed only once.
- So, if dedicated field would be present, then sending it each time would not be required.
- For this reason, MSS is informed once using Options field.

Thank You