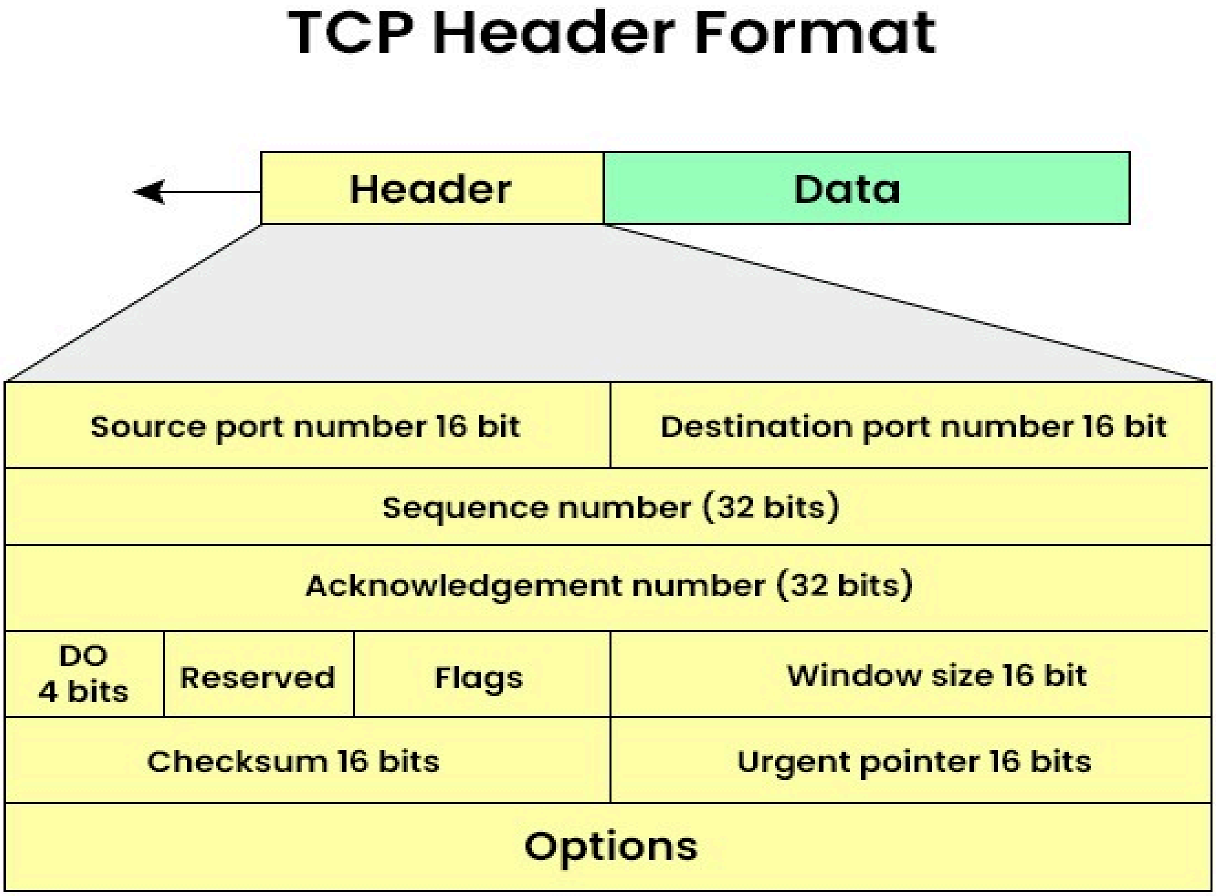
TCP HEADER


TCP Header Format
Header
Data
Source port number 16 bit
Destination port number 16 bit
Sequence number (32 bits)
Acknowledgement number (32 bits)
DO 4 bits
Reserved
Flags
Window size 16 bit
Checksum 16 bits
Urgent pointer 16 bits
Options

1. Introduction to TCP and its Header Format

- **TCP (Transmission Control Protocol)**: A reliable, connection-oriented protocol used for data transmission between devices in a network.
- **TCP Header Fields**:
  - **Source Port**: Identifies the sending port.
  - **Destination Port**: Identifies the receiving port.
  - **Sequence Number**: Used to keep track of data segments.
  - **Acknowledgment Number**: Confirms receipt of data.
  - **Data Offset**: Size of the TCP header.
  - **Flags**: Control bits (e.g., SYN, ACK, FIN).
  - **Window**: Specifies the size of the receiver's buffer (advertised window).
  - **Checksum**: Error-checking for the header and data.
  - **Urgent Pointer**: Indicates if any data is urgent.
  - **Options**: Used for various purposes, including MSS (Maximum Segment Size).

2. TCP Connection Establishment (Three-Way Handshake)

- **Objective**: Establish a reliable connection between the sender and receiver, and agree on the parameters such as MSS and the advertisement window.

Step-by-Step Process:

- **Step 1: SYN (Synchronize)**
    - **Sender (Client)**: Chooses an Initial Sequence Number (ISN), say 1000, and sends a TCP segment with the SYN flag set. The sender also advertises its MSS (Maximum Segment Size) and its initial advertised window size.
    - **TCP Header Example**:
        - **Sequence Number**: 1000
        - **Flags**: SYN=1, ACK=0
        - **Options**: MSS=1460 bytes (assuming a typical Ethernet MTU minus TCP/IP headers)
        - **Window Size**: 5000 bytes (advertised window)
    - **Purpose**: The sender indicates it wants to establish a connection, synchronize sequence numbers, and informs the receiver of the largest segment size it can receive as well as its buffer capacity.
- **Step 2: SYN-ACK (Synchronize-Acknowledge)**
    - **Receiver (Server)**: Chooses its own ISN, say 5000, and sends a TCP segment with the SYN and ACK flags set. The Acknowledgment Number is set to the sender's ISN + 1. The receiver also advertises its MSS and window size.
    - **TCP Header Example**:
        - **Sequence Number**: 5000
        - **Acknowledgment Number**: 1001
        - **Flags**: SYN=1, ACK=1
        - **Options**: MSS=1460 bytes
        - **Window Size**: 8000 bytes (advertised window)
    - **Purpose**: The receiver acknowledges the sender's request, provides its own sequence number for synchronization, and shares its MSS and buffer size for data transfer.
- **Step 3: ACK (Acknowledge)**
    - **Sender (Client)**: Sends a TCP segment with the ACK flag set and the Acknowledgment Number set to the receiver's ISN + 1, completing the connection establishment.
    - **TCP Header Example**:
        - **Sequence Number**: 1001
        - **Acknowledgment Number**: 5001
        - **Flags**: ACK=1
        - **Window Size**: 5000 bytes (advertised window)
    - **Purpose**: The sender acknowledges the receiver's sequence number and confirms the connection establishment.

Flowchart:

```
Client (Sender)                        Server (Receiver)
------------------------------------------------------------------
| SYN, Seq = 1000, MSS=1460, Window=5000    |                    |
|------------------------------------------>|                    |
|                                  | SYN-ACK, Seq = 5000, Ack = 1001, |
|                                  | MSS=1460, Window=8000           |
|                                  |<----------------------------------- |
| ACK, Seq = 1001, Ack = 5001, Window=5000 |                     |
|------------------------------------------>|                    |
------------------------------------------------------------------
Connection Established
```

3. Data Transfer Phase

- **Objective**: Exchange data between the sender and receiver using the established connection while respecting the negotiated MSS and window sizes.

Example Scenario:

- **Sender (Client)** sends data with a sequence number of 1001 and a length of 100 bytes, ensuring that the data size does not exceed the MSS (1460 bytes).
- **Receiver (Server)** acknowledges the data by sending an acknowledgment with an acknowledgment number of 1101 (1001 + 100) and updates the advertised window size based on how much buffer space is available.

TCP Header Example for Data Transmission:

- **Client to Server**:
  - **Sequence Number**: 1001
  - **Data Length**: 100 bytes
  - **Flags**: ACK=1
  - **MSS**: Indicated in options (1460 bytes).
  - **Window Size**: 4900 bytes (remaining buffer after 100 bytes are sent)
- **Server to Client**:
  - **Acknowledgment Number**: 1101
  - **Window Size**: 7900 bytes (remaining buffer after processing 100 bytes)

```
Client (Sender)                          Server (Receiver)
------------------------------------------------------------------
| Data, Seq = 1001, Len = 100, Window=4900  |                      |
|----------------------------------------->|                       |
|                                          | ACK, Ack = 1101,
Window=7900        |
|
|<---------------------------------- |
| Data continues...                        |                       |
------------------------------------------------------------------
```

4. Connection Termination (Four-Way Handshake)

- **Objective**: Gracefully close the connection after data transfer is complete.

Step-by-Step Process:

- **Step 1: FIN (Finish)**
    - **Sender (Client)**: Sends a TCP segment with the FIN flag set, indicating it has finished sending data.
    - **TCP Header Example**:
        - **Sequence Number**: 1101
        - **Flags**: FIN=1, ACK=1
        - **Window Size**: 4900 bytes (advertised window)
- **Step 2: ACK (Acknowledge)**
    - **Receiver (Server)**: Acknowledges the FIN request by sending a TCP segment with the ACK flag set.
    - **TCP Header Example**:
        - **Sequence Number**: 5001
        - **Acknowledgment Number**: 1102
        - **Flags**: ACK=1
        - **Window Size**: 7900 bytes (advertised window)
- **Step 3: FIN (Finish)**
    - **Receiver (Server)**: Sends a TCP segment with the FIN flag set, indicating it has finished sending data.
    - **TCP Header Example**:
        - **Sequence Number**: 5001
        - **Flags**: FIN=1, ACK=1
        - **Window Size**: 7900 bytes (advertised window)
- **Step 4: ACK (Acknowledge)**
    - **Sender (Client)**: Acknowledges the server's FIN request by sending a TCP segment with the ACK flag set.
    - **TCP Header Example**:
        - **Sequence Number**: 1102
        - **Acknowledgment Number**: 5002
        - **Flags**: ACK=1

        - **Window Size**: 4900 bytes (advertised window)

```
Client (Sender)                         Server (Receiver)
------------------------------------------------------------------
| FIN, Seq = 1101, Window=4900           |                      |
|----------------------------------------->|                      |
|                                         | ACK, Ack = 1102,
Window=7900       |
|
|<---------------------------------- |
|                                         | FIN, Seq = 5001,
Window=7900      |
|
|<---------------------------------- |
| ACK, Ack = 5002, Window=4900            |                      |
|----------------------------------------->|                      |
------------------------------------------------------------------
Connection Closed
```

5. Conclusion

- **Key Points**:
    - The three-way handshake establishes a reliable connection and negotiates important parameters like MSS and the advertised window size.
    - Data transfer uses sequence and acknowledgment numbers to ensure all data is received correctly, with the MSS and window sizes controlling the flow.
    - The connection is terminated gracefully using a four-way handshake.