**TCP (Transmission Control Protocol)**

* Connection oriented, it establishes connection using a three-way handshake
* Guaranteed data delivery. TCP ensures that data is delivered accurately and in the same order it was sent. If any data is lost, it is retransmitted
* TCP uses checksums to check for errors in the transmitted data. If errors are detected, the corrupted segments are discarded and retransmitted.
* TCP manages the rate of data transmission between a sender and a receiver to prevent a fast sender from overwhelming a slow receiver. This is achieved using a mechanism called sliding window.
* TCP includes mechanisms to detect network congestion and reduce the rate of data transmission to alleviate congestion. Techniques like slow start, congestion avoidance, fast retransmit, and fast recovery are used.

**The Three-Way Handshake**

Before data transfer begins, TCP uses a three-step process to establish a connection:

1. **SYN**: The client sends a SYN (synchronize) packet to the server, initiating the connection.
2. **SYN-ACK**: The server responds with a SYN-ACK (synchronize-acknowledge) packet, acknowledging the client's request and establishing its own.
3. **ACK**: The client sends an ACK (acknowledge) packet back to the server, completing the connection setup.

**Data Transfer**

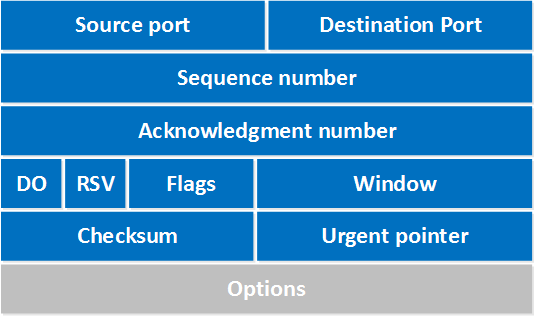
Once the connection is established, data can be sent in segments. Each segment contains a sequence number so the receiver can reorder segments if they arrive out of order. The receiver sends back acknowledgment (ACK) packets to confirm the receipt of segments.

**Common Applications of TCP**

* **Web Browsing**: HTTP and HTTPS, the protocols used for web browsing, run on top of TCP.
* **Email**: Protocols like SMTP, POP3, and IMAP use TCP to ensure emails are sent and received reliably.
* **File Transfer**: FTP (File Transfer Protocol) and SFTP (Secure File Transfer Protocol) rely on TCP for reliable file transfers.
* **Remote Access**: Protocols like SSH (Secure Shell) and Telnet use TCP to provide secure and reliable remote access to devices.

**TCP segment**

Segment is the data that is attached with the header.



TCP header

* **Source port**: this is a 16 bit field that specifies the port number of the sender.
* **Destination port**: this is a 16 bit field that specifies the port number of the receiver.
* **Sequence number**: the sequence number is a 32 bit field that indicates how much data is sent during the TCP session. When you establish a new TCP connection (3 way handshake) then the initial sequence number is a random 32 bit value. The receiver will use this sequence number and sends back an acknowledgment. Protocol analyzers like wireshark will often use a *relative sequence number of 0* since it’s easier to read than some high random number.
* **Acknowledgment number**: this 32 bit field is used by the receiver to request the next TCP segment. This value will be the sequence number incremented by 1.
* **DO**: this is the 4 bit data offset field, also known as the header length. It indicates the length of the TCP header so that we know where the actual data begins.
* **RSV**: these are 3 bits for the reserved field. They are unused and are always set to 0.
* **Flags**: there are 9 bits for flags, we also call them control bits. We use them to establish connections, send data and terminate connections:
  + **URG**: urgent pointer. When this bit is set, the data should be treated as priority over other data.
  + **ACK**: used for the acknowledgment.
  + **PSH**: this is the push function. This tells an application that the data should be transmitted immediately and that we don’t want to wait to fill the entire TCP segment.
  + **RST**: this resets the connection, when you receive this you have to terminate the connection right away. This is only used when there are unrecoverable errors and it’s not a normal way to finish the TCP connection.
  + **SYN**: we use this for the initial three way handshake and it’s used to set the initial sequence number.
  + **FIN**: this finish bit is used to end the TCP connection. TCP is full duplex so both parties will have to use the FIN bit to end the connection. This is the normal method how we end an connection.
* **Window**: the 16 bit window field specifies how many bytes the receiver is willing to receive. It is used so the receiver can tell the sender that it would like to receive more data than what it is currently receiving. It does so by specifying the number of bytes beyond the sequence number in the acknowledgment field.
* **Checksum**: 16 bits are used for a checksum to check if the TCP header is OK or not.
* **Urgent pointer**: these 16 bits are used when the URG bit has been set, the urgent pointer is used to indicate where the urgent data ends.
* **Options**: this field is optional and can be anywhere between 0 and 320 bits.

**User Datagram Protocol (UDP)**

UDP is another core protocol of the Internet Protocol (IP) suite, which is used for transmitting data. Unlike TCP, UDP is a connectionless protocol, meaning it does not establish a connection before sending data and does not guarantee the delivery, order, or integrity of packets. Here are the main characteristics and uses of UDP:

**Key Characteristics of UDP:**

1. **Connectionless Protocol**: UDP does not establish a connection before data is sent. Each packet is sent independently, without any acknowledgment or guarantee of delivery.
2. **No Error Checking or Correction**: UDP does not provide mechanisms for error checking or correction. If packets are lost or corrupted, they are not retransmitted.
3. **Minimal Overhead**: Because UDP lacks the error checking and connection establishment features of TCP, it has lower overhead, making it faster and more efficient for certain applications.
4. **No Flow Control or Congestion Control**: UDP does not manage the rate of data transmission or avoid network congestion, relying on the application layer to handle these issues if necessary.
5. **Datagram-Based**: UDP sends data in discrete packets called datagrams. Each datagram is sent independently and may take different paths to reach the destination.

**Common Applications of UDP:**

* **Streaming Media**: UDP is often used for live video and audio streaming, where timely delivery is more critical than perfect reliability. Examples include IPTV and live broadcasts.
* **Online Gaming**: Many online multiplayer games use UDP because it provides low latency communication, which is crucial for real-time interaction.
* **Voice over IP (VoIP)**: VoIP services like Skype use UDP to transmit voice data, as the protocol’s lower latency is beneficial for real-time voice communication.
* **DNS (Domain Name System)**: DNS queries and responses are typically transmitted using UDP due to the small size and simple structure of the requests and responses.
* **TFTP (Trivial File Transfer Protocol)**: TFTP uses UDP for simple, low-overhead file transfers, often in bootstrapping or network management tasks.

**UDP header**

