Cookies, DNS and TCP

Agenda

- What are cookies, and how do they work?
- What is DNS, how is it structured?
- TCP vs UDP
    - Headers
    - 3 way handshake

Key terms

Cookies

> They are small blocks of data created by a web server while a user is browsing a website and placed on the user's computer or other device by the user's web browser.

DNS

> The Domain Name System is the hierarchical and decentralized naming system used to identify computers reachable through the Internet or other Internet Protocol networks.

TCP

> TCP stands for Transmission Control Protocol a communications standard that enables application programs and computing devices to exchange messages over a network

UDP

> The User Datagram Protocol, or UDP, is a communication protocol used across the Internet for especially time-sensitive transmissions such as video playback or DNS lookups. It speeds up communications by not formally establishing a connection before data is transferred

3-way handshake

> The 3-Way Handshake process is the defined set of steps that takes place in the TCP for creating a secure and reliable communication link and also closing it

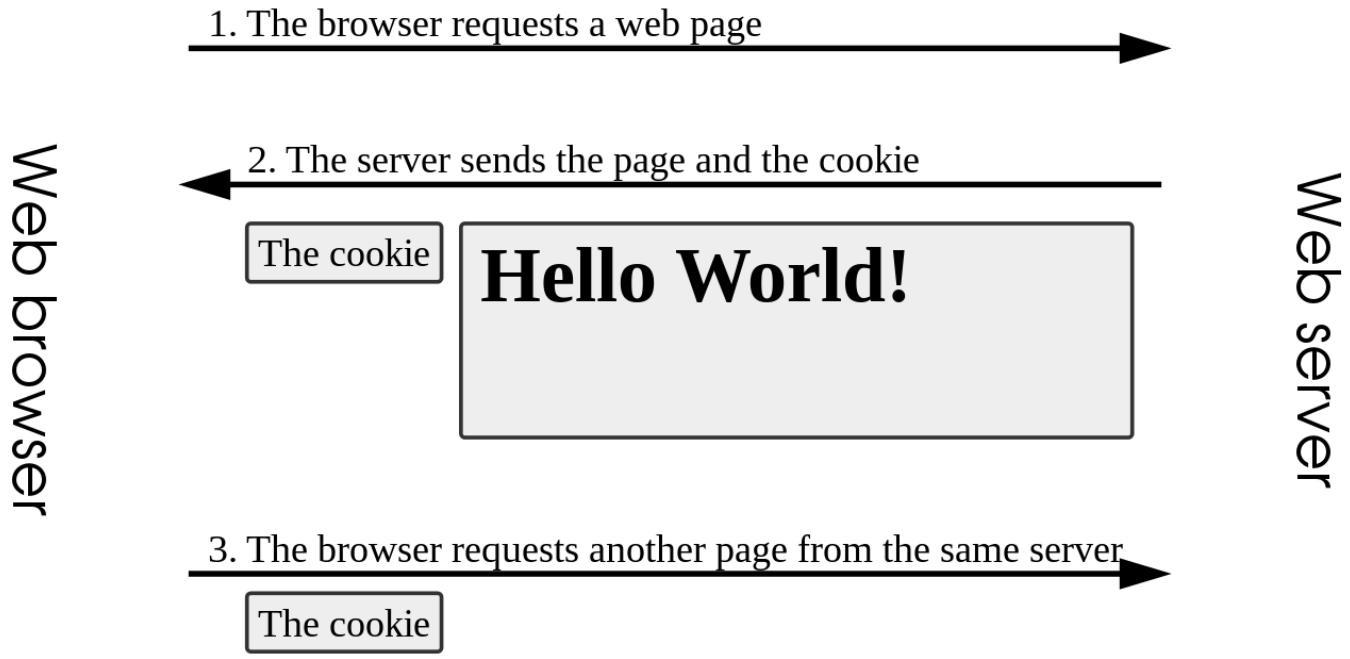
Cookies

HTTP is a stateless protocol, meaning that the server does not maintain any state. To provide a better user experience, such as remembering if user is logged in, state is required to be maintained. Cookies are a way to achieve state in HTTP request and response cycles.

The server is responsible for creation of the cookies. A web server specifies a cookie to be stored by sending an HTTP header called `Set-Cookie`.

When a cookie is present, and the optional rules allow, the cookie value is sent to the server with each subsequent request. The cookie value is stored in an HTTP header called Cookie and contains just the

cookie value without any of the other options.



Request and response lifecycle

1. The browser sends its first HTTP request for the homepage of the `www.slow.com` website

```
GET /index.html HTTP/1.1
Host: www.slow.com
...
```

2. The server responds with the data and `Set-Cookie` headers field

```
HTTP/1.0 200 OK
Content-type: text/html
Set-Cookie: theme=light
Set-Cookie: sessionToken=abc123; Expires=Wed, 09 Jun 2021 10:18:14 GMT
...
```

   The server's HTTP response contains the contents of the website's homepage. But it also instructs the browser to set two cookies. The first, `theme`, is considered to be a session cookie since it does not have an Expires or Max-Age attribute. Session cookies are intended to be deleted by the browser when the browser closes.

   The second, `sessionToken`, is considered to be a persistent cookie since it contains an Expires attribute, which instructs the browser to delete the cookie at a specific date and time

3. Next, the browser sends another request to visit the spec.html page on the website. This request contains a Cookie header field, which contains the two cookies that the server instructed the browser

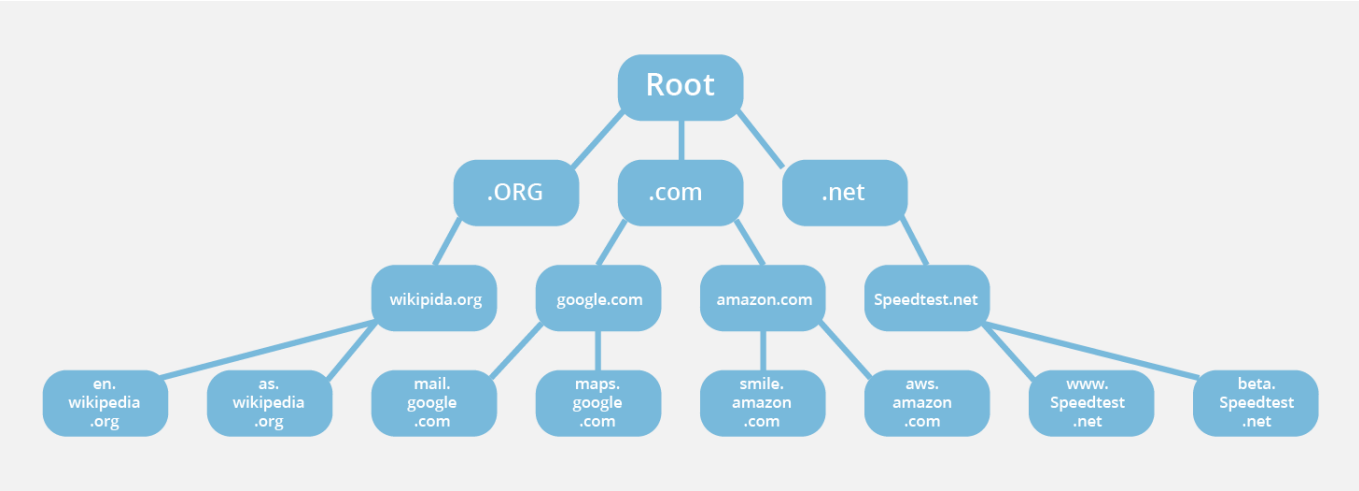
to set:

```
GET /spec.html HTTP/1.1
Host: www.slow.com
Cookie: theme=light; sessionToken=abc123
…
```

This way, the server knows that this HTTP request is related to the previous one.

DNS

> The Domain Name System (DNS) is the phonebook of the Internet. Humans access information online through domain names, like nytimes.com or espn.com. Web browsers interact through Internet Protocol (IP) addresses. DNS translates domain names to IP addresses so browsers can load Internet resources.



DNS is a hierarchical system that organizes the Internet by domain name. There are four DNS servers involved in the process:

Root DNS server

The administration of the Domain Name System (DNS) is structured in a hierarchy using different managed areas or “zones”, with the root zone at the very top of that hierarchy. Root servers are DNS nameservers that operate in the root zone. These servers can directly answer queries for records stored or cached within the root zone.

Top-level DNS server

A TLD nameserver maintains information for all the domain names that share a common domain extension, such as .com, .net, or whatever comes after the last dot in a url. For example, a .com TLD nameserver contains information for every website that ends in ‘.com’. If a user was searching for google.com, after receiving a response from a root nameserver, the recursive resolver would then send a query to a .com TLD nameserver, which would respond by pointing to the authoritative nameserver for that domain.

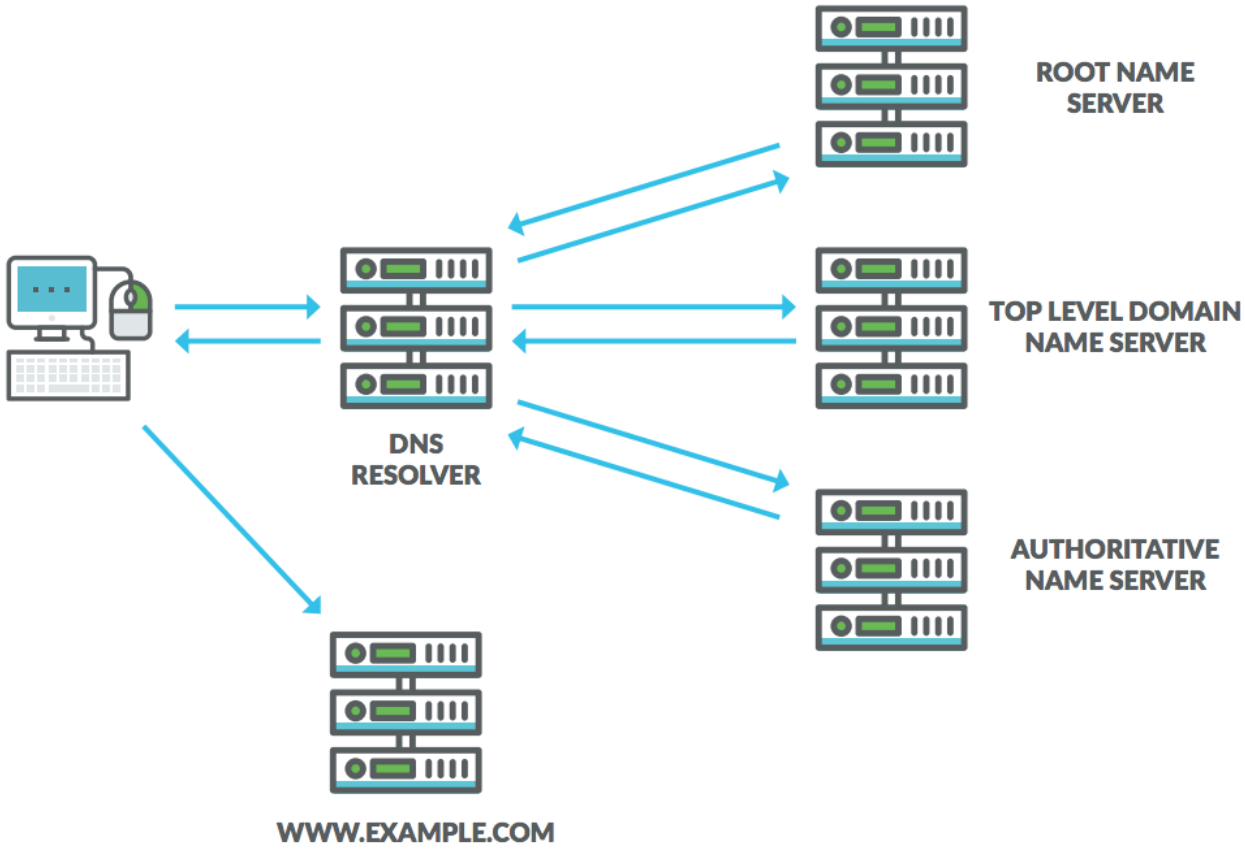
Authoritative nameserver

This final nameserver can be thought of as a dictionary on a rack of books, in which a specific name can be translated into its definition. The authoritative nameserver is the last stop in the nameserver query. If the authoritative name server has access to the requested record, it will return the IP address for the requested hostname back to the DNS Recursor (the librarian) that made the initial request.

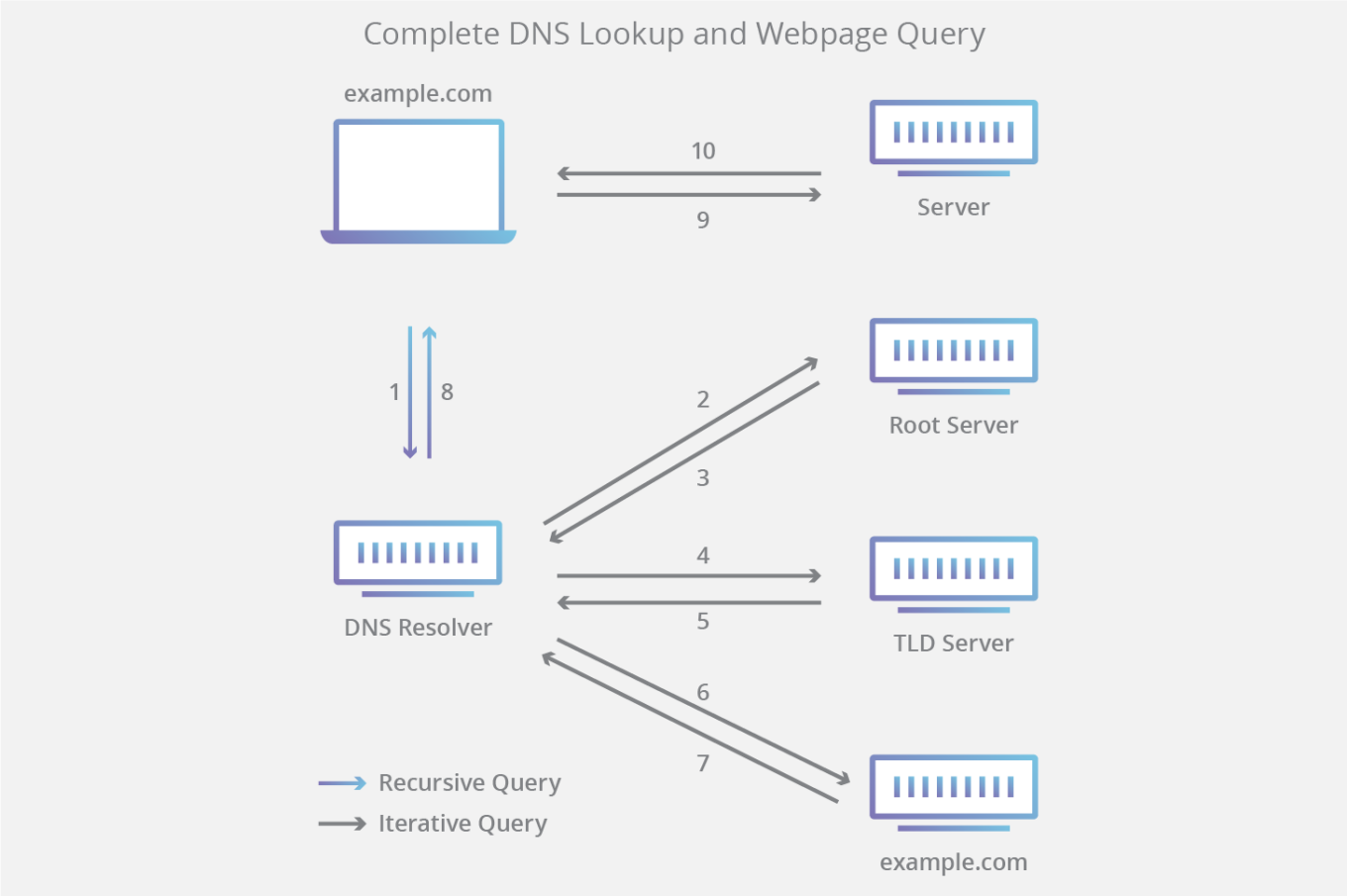
The authoritative nameserver is usually the resolver's last step in the journey for an IP address. The authoritative nameserver contains information specific to the domain name it serves (e.g. google.com) and it can provide a recursive resolver with the IP address of that server.

Recursive resolver

The recursor can be thought of as a librarian who is asked to go find a particular book somewhere in a library. The DNS recursor is a server designed to receive queries from client machines through applications such as web browsers. Typically the recursor is then responsible for making additional requests in order to satisfy the client's DNS query.



Steps in the DNS query process



1. A user types `slow.com` into a web browser and the query travels into the Internet and is received by a DNS recursive resolver.
2. The resolver then queries a DNS root nameserver (.).
3. The root server then responds to the resolver with the address of a Top Level Domain (TLD) DNS server (such as .com or .net), which stores the information for its domains. When searching for slow.com, our request is pointed toward the .com TLD.
4. The resolver then makes a request to the .com TLD.
5. The TLD server then responds with the IP address of the domain's nameserver, slow.com.
6. Lastly, the recursive resolver sends a query to the domain's nameserver.
7. The IP address for slow.com is then returned to the resolver from the nameserver.
8. The DNS resolver then responds to the web browser with the IP address of the domain requested initially.

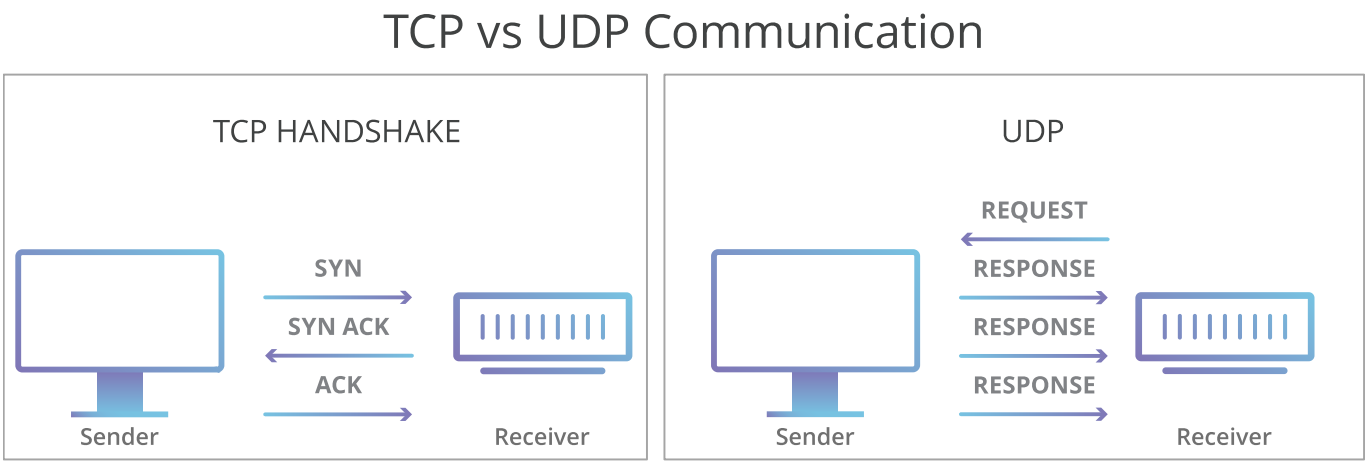
TCP and UDP

UDP

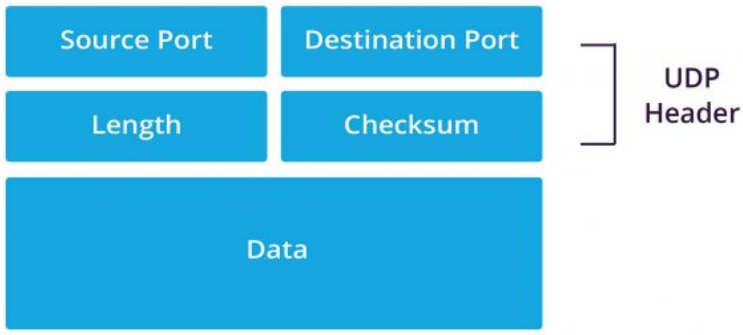
UDP is a standardized method for transferring data between two computers in a network. Compared to other protocols, UDP accomplishes this process in a simple fashion: it sends packets (units of data transmission) directly to a target computer, without establishing a connection first, indicating the order of said packets, or checking whether they arrived as intended. (UDP packets are referred to as 'datagrams'.)

UDP uses a simple connectionless communication model with a minimum of protocol mechanisms. UDP provides checksums for data integrity, and port numbers for addressing different functions at the source and destination of the datagram. It has no handshaking dialogues, and thus exposes the user's program to

any unreliability of the underlying network; there is no guarantee of delivery, ordering, or duplicate protection.



UDP header



UDP wraps datagrams with a UDP header, which contains four fields totalling eight bytes. The fields in a UDP header are:

- `Source port` – The port of the device sending the data. This field can be set to zero if the destination computer doesn't need to reply to the sender.
- `Destination port` – The port of the device receiving the data. UDP port numbers can be between 0 and 65,535.
- `Length` – Specifies the number of bytes comprising the UDP header and the UDP payload data.
- `Checksum` – The checksum allows the receiving device to verify the integrity of the packet header and payload.

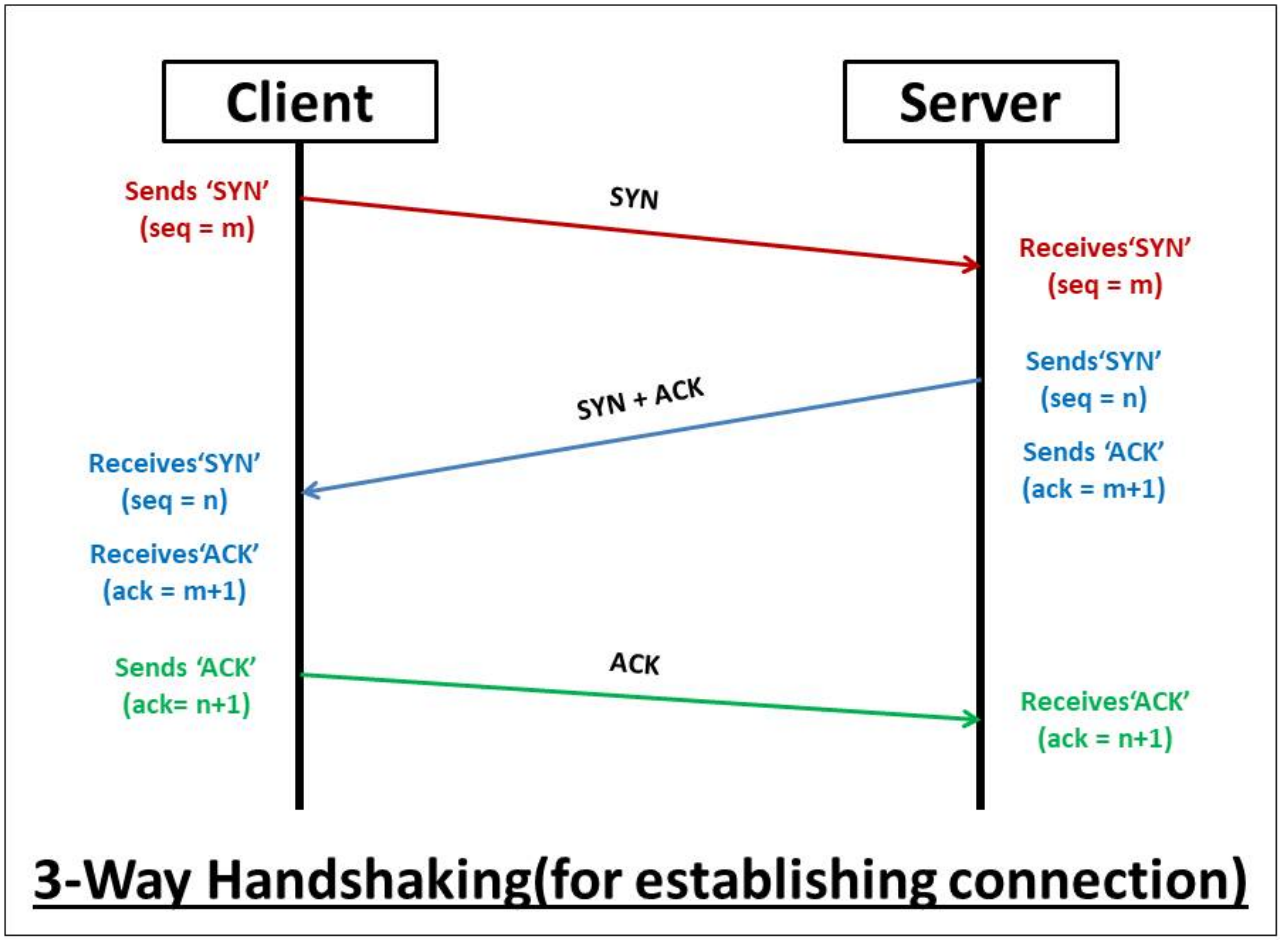
TCP

TCP provides reliable, ordered, and error-checked delivery of a stream of octets (bytes) between applications running on hosts communicating via an IP network

TCP is connection-oriented, and a connection between client and server is established before data can be sent. The server must be listening (passive open) for connection requests from clients before a connection

is established. Three-way handshake (active open), retransmission, and error detection adds to reliability but lengthens latency.

Connection establishment

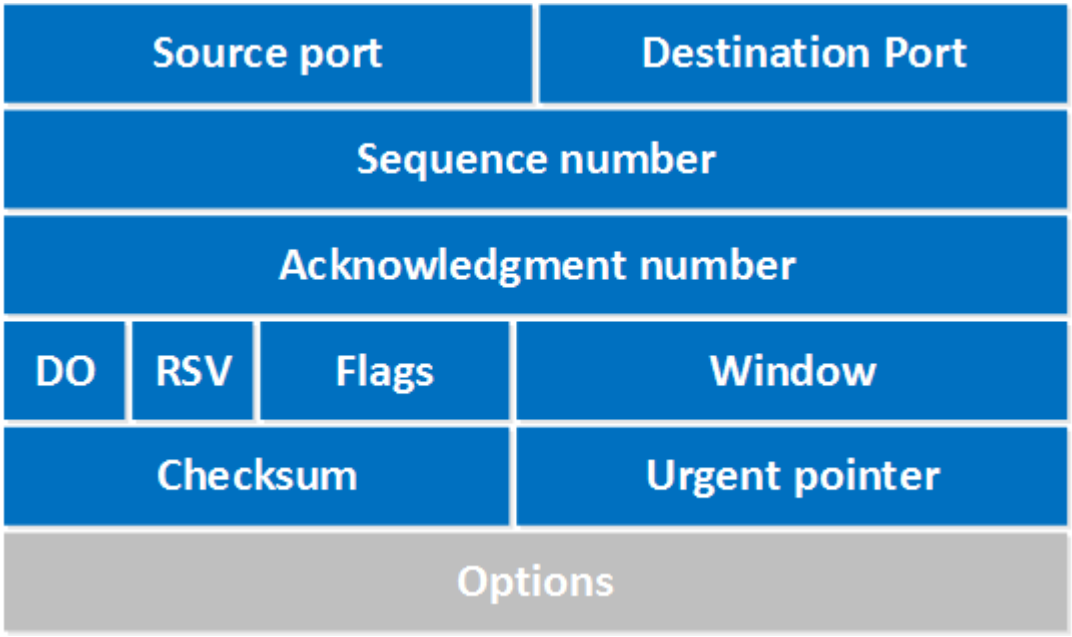


Before a client attempts to connect with a server, the server must first bind to and listen at a port to open it up for connections: this is called a passive open. Once the passive open is established, a client may establish a connection by initiating an active open using the three-way (or 3-step) handshake:

1. SYN: The active open is performed by the client sending a SYN to the server. The client sets the segment's sequence number to a random value A.
2. SYN–ACK: In response, the server replies with a SYN-ACK. The acknowledgment number is set to one more than the received sequence number i.e. A+1, and the sequence number that the server chooses for the packet is another random number, B.
3. ACK: Finally, the client sends an ACK back to the server. The sequence number is set to the received acknowledgment value i.e. A+1, and the acknowledgment number is set to one more than the received sequence number i.e. B+1.

Steps 1 and 2 establish and acknowledge the sequence number for one direction. Steps 2 and 3 establish and acknowledge the sequence number for the other direction. Following the completion of these steps, both the client and server have received acknowledgments and a full-duplex communication is established

TCP header



The fields in a TCP header are:

- `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.
- 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.

TCP vs UDP

Feature	TCP	UDP
Connection status	Requires an established connection to transmit data (connection should be closed once transmission is complete)	Connectionless protocol with no requirements for opening, maintaining, or terminating a connection
Data sequencing	Able to sequence	Unable to sequence
Guaranteed delivery	Can guarantee delivery of data to the destination router	Cannot guarantee delivery of data to the destination
Retransmission of data	Retransmission of lost packets is possible	No retransmission of lost packets
Error checking	Extensive error checking and acknowledgment of data	Basic error checking mechanism using checksums
Method of transfer	Data is read as a byte stream; messages are transmitted to segment boundaries	UDP packets with defined boundaries; sent individually and checked for integrity on arrival
Speed	Slower than UDP	Faster than TCP
Broadcasting	Does not support Broadcasting	Does support Broadcasting
Optimal use	Used by HTTPS, HTTP, SMTP, POP, FTP, etc	Video conferencing, streaming, DNS, VoIP, etc

Reading list

- Cookies in depth
- Sliding Window