Programming and Communications III: HTTPv2

Jordi Ricard Onrubia Palacios

Departament d'Informàtica i Enginyeria Industrial Universitat de Lleida

Universitat de Lleida
Departament d'Informàtica
i Enginyeria Industrial

Programming and Communications III: HTTPv2 Overview

HTTPv2 Overview

- HTTP (HyperText Transfer Protocol) is an application protocol designed for transmission of hypertext documents on Internet
- Ports: 80/TCP, 443/TCP (8000, 8080, 81, etc)
- Location in the protocol stack
  - Application: HTTP
  - Transport: TCP
  - Network: IP
- Client / Server communication
- It is a stateless protocol

Universitat de Lleida
Departament d'Informàtica i Enginyeria Industrial

Programming and Communications III: What are Ports

What are Ports

- In computer networking, ports are logical endpoints used for communication between devices.
- They allow a single device (with one IP address) to manage multiple simultaneous connections by distinguishing between different types of network traffic.
- Ports are represented by 16-bit numbers, so the range is 0–65535
  - Well-Known Ports (0–1023): Reserved for common services and protocols.
  - Registered Ports (1024–49151): Assigned to specific services or applications by organizations but not reserved.
  - Dynamic or Private Ports (49152–65535): Used for ephemeral (temporary) connections.

Universitat de Lleida
Departament d'Informàtica
i Enginyeria Industrial

Programming and Communications III: What is TCP

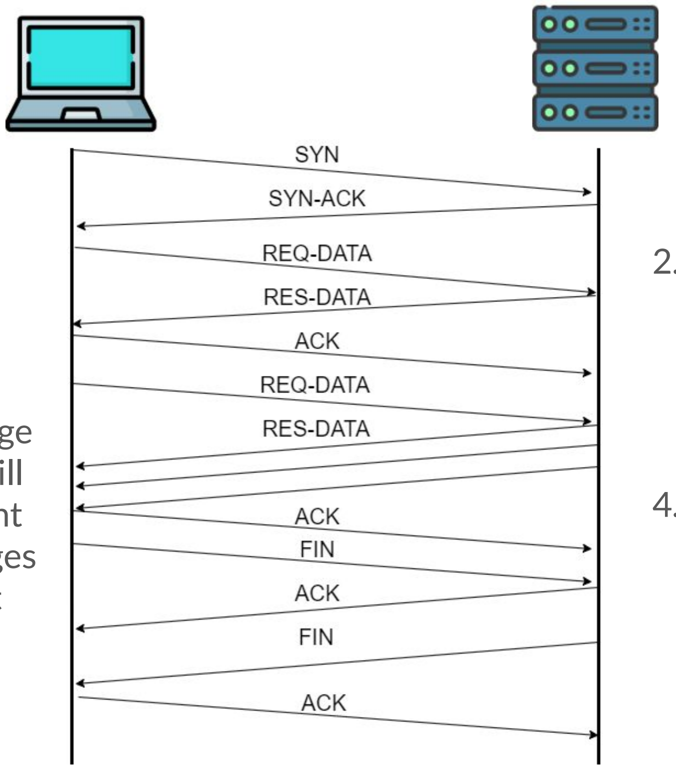
What is TCP

- TCP is a connection-oriented protocol used for reliable communication between devices on a network.
- Reliable Data Transfer:
  - Ensures data is delivered accurately and in order.
  - Uses error detection and retransmission for reliability.
- Connection-Oriented: Requires a connection to be established before data transfer (via the three-way handshake).

What is TCP

- Stream-Oriented: Treats data as a continuous stream, breaking it into manageable segments.
- Flow Control: Ensures the sender does not overwhelm the receiver by using a sliding window mechanism.
- Congestion Control: Adjusts the rate of data transmission to avoid network congestion.
- Error Checking: Uses checksums to verify data integrity during transmission.
- Full-Duplex Communication: Data can flow simultaneously in both directions.

What is TCP



1. Establish Communication, Client sends ACK, Server answers with SYN-ACK

2. Client Requests Data, Server Responses with the Data, once the client has received all the Data sends an ACK

3. If the client Requests for a large quantity of data, the server will chunk it and send it in different packages. Once all the packages have been received, the client will send an ACK

4. To finish the connection the Client sends a FIN, the server then first sends an ACK to acknowledge the FIN message and then sends a FIN, once the Client acknowledges the FIN messages the connection ends

Universitat de Lleida
Departament d'Informàtica i Enginyeria Industrial

Programming and Communications III: Client/Server Architecture

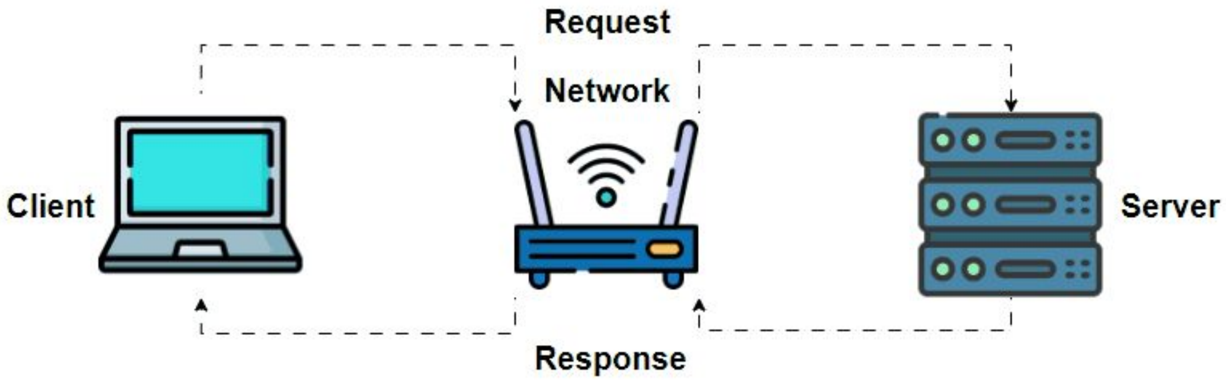
Client/Server Architecture

- A client-server architecture is a network design where tasks or workloads are divided between:
    - Clients: Request services.
    - Servers: Provide services.
- Components
    - Client: A device or application that initiates requests (Web browsers, mobile apps, or a desktop computer).
    - Server: A central machine or application that processes requests and provides responses (Web servers, database servers, or file servers).
    - Network: Connects clients and servers, typically over the internet or a local network.

Client/Server Architecture

Example:

1. Request: The client sends a request to the server (e.g., requesting a web page).
2. Processing: The server processes the request (e.g., retrieves the web page or queries a database).
3. Response: The server sends the processed result back to the client.



Universitat de Lleida
Departament d'Informàtica
i Enginyeria Industrial

Programming and Communications III: Connection Lifecycle

HTTP Connection Lifecycle

1. Client Initiates a Request:
   1.1. The client (e.g., browser) sends a request to a server, typically through a URL.
   1.2. The request specifies:
      1.2.1. Method: (e.g., GET, POST, PUT, DELETE).
      1.2.2. Headers: Metadata about the request (e.g., User-Agent, Accept, etc.).
      1.2.3. Body (optional): Data sent in the request, e.g., form data for POST requests.

HTTP Connection Lifecycle

2. DNS Lookup:
   2.1. The URL's domain name is resolved to an IP address using DNS (Domain Name System).
3. TCP Connection:
   3.1. The client establishes a TCP connection to the server (default port is 80 for HTTP, 443 for HTTPS).
   3.2. In HTTPS, an additional TLS/SSL handshake ensures encryption and secure communication.

HTTP Connection Lifecycle

4. Server Processes the Request:
   - 4.1. The server interprets the request, fetches the resource, or performs the required operation.
5. Server Sends a Response:
   - 5.1. The server sends back:
     - 5.1.1. Status Code: Indicates success or error (e.g., 200 OK, 404 Not Found).
     - 5.1.2. Headers: Metadata about the response (e.g., Content-Type, Cache-Control).
     - 5.1.3. Body (optional): The actual content (e.g., HTML, JSON, or an image).

HTTP Connection Lifecycle

6. Client Renders the Response: The browser or application processes the response and displays it to the user.

Universitat de Lleida
Departament d'Informàtica
i Enginyeria Industrial

Programming and Communications III: HTTP Methods

HTTP Methods

HTTP methods are verbs used in the HTTP to specify the desired action to be performed on a resource (like a web page, data, or API endpoint).

Providing:

- Uniformity: They standardize communication between clients and servers.
- Clarity: Clearly define what operation is expected (read, create, update, delete).
- Security: Proper method usage helps prevent unintended side effects.
- REST APIs: HTTP methods are foundational for designing RESTful APIs.

HTTP Methods

GET

- Purpose: Retrieve data from a server.
- Characteristics:
  - Does not modify server data.
  - Safe and idempotent.
  - Can be cached.
- Example: Fetching a webpage or retrieving a list of items.

POST

- Purpose: Send data to a server to create a resource.
- Characteristics:
  - May modify server data.
  - Not idempotent.
  - Often used with forms or APIs for creating new entries.
- Example: Submitting a form to create a new user.

HTTP Methods

PUT

- Purpose: Update or replace an existing resource.
- Characteristics:
  - Idempotent (sending the same request multiple times results in the same outcome).
  - Requires the full resource data in the request body.
- Example: Updating a user's profile information.

PATCH

- Purpose: Partially update an existing resource.
- Characteristics:
  - Idempotent.
  - Only the changes are sent in the request body.
- Example: Changing a user's email address without modifying other profile details.

HTTP Methods

DELETE

- Purpose: Remove a resource from the server.
- Characteristics:
  - Idempotent.
  - Deletes the specified resource.
- Example: Removing a user from a database.

Universitat de Lleida
Departament d'Informàtica
i Enginyeria Industrial

Programming and Communications III: HTTP Status Codes

HTTP Status Codes

HTTP status codes are three-digit numbers returned by a web server to indicate the outcome of a client's request. They help communicate whether the request was successful, encountered an error, or requires additional steps. These codes are a critical part of the HTTP protocol, providing standardized responses to client applications like web browsers, mobile apps, or APIs.

HTTP Status Codes

1xx: Informational Responses

These codes indicate that a request was received and is being processed.

- 100 Continue: The server has received the request headers and the client can proceed with the request body.
- 101 Switching Protocols: The server agrees to switch to a different protocol as requested by the client.
- 102 Processing: The server has received and is processing the request, but no response is available yet.

HTTP Status Codes

<u>2xx: Success</u>

These codes mean the request was successfully received, understood, and accepted.

- 200 OK: The request was successful, and the response contains the requested data.
- 201 Created: A new resource has been created as a result of the request.
- 202 Accepted: The request has been accepted for processing but is not yet completed.
- 204 No Content: The request was successful, but there is no content to send back.

HTTP Status Codes

3xx: Redirection

These codes indicate the client must take additional actions to complete the request.

- 301 Moved Permanently: The resource has been permanently moved to a new URL.
- 302 Found: The resource is temporarily available at a different URL.
- 303 See Other: The client should use a GET request to fetch the resource at another URL.
- 304 Not Modified: The resource hasn't changed since the last request; the client can use cached data.
- 307 Temporary Redirect: The resource is temporarily moved, and the same HTTP method should be used.
- 308 Permanent Redirect: Similar to 301 but mandates the original HTTP method.

HTTP Status Codes

4xx: Client Errors

These codes indicate issues with the client's request.

- 400 Bad Request: The server couldn't understand the request due to invalid syntax.
- 401 Unauthorized: Authentication is required but has not been provided or is invalid.
- 403 Forbidden: The client doesn't have permission to access the resource.
- 404 Not Found: The requested resource couldn't be found.
- 405 Method Not Allowed: The HTTP method used is not supported for the resource.
- 409 Conflict: The request could not be processed due to a conflict with the current state of the resource.
- 410 Gone: The resource is no longer available and has been permanently removed.
- 429 Too Many Requests: The client has sent too many requests in a given time period.

HTTP Status Codes

5xx: Server Errors

These codes indicate the server failed to fulfill a valid request.

- 500 Internal Server Error: The server encountered an unexpected condition that prevented it from fulfilling the request.
- 501 Not Implemented: The server doesn't support the functionality required to fulfill the request.
- 502 Bad Gateway: The server received an invalid response from an upstream server.
- 503 Service Unavailable: The server is temporarily unable to handle the request, often due to maintenance or overload.
- 504 Gateway Timeout: The server didn't receive a timely response from an upstream server.
- 505 HTTP Version Not Supported: The server doesn't support the HTTP protocol version used in the request.