

COMPUTER NETWORKS AND COMMUNICATION PROTOCOLS

Web Access: HTTP

16501018

Mehmet KORKMAZ

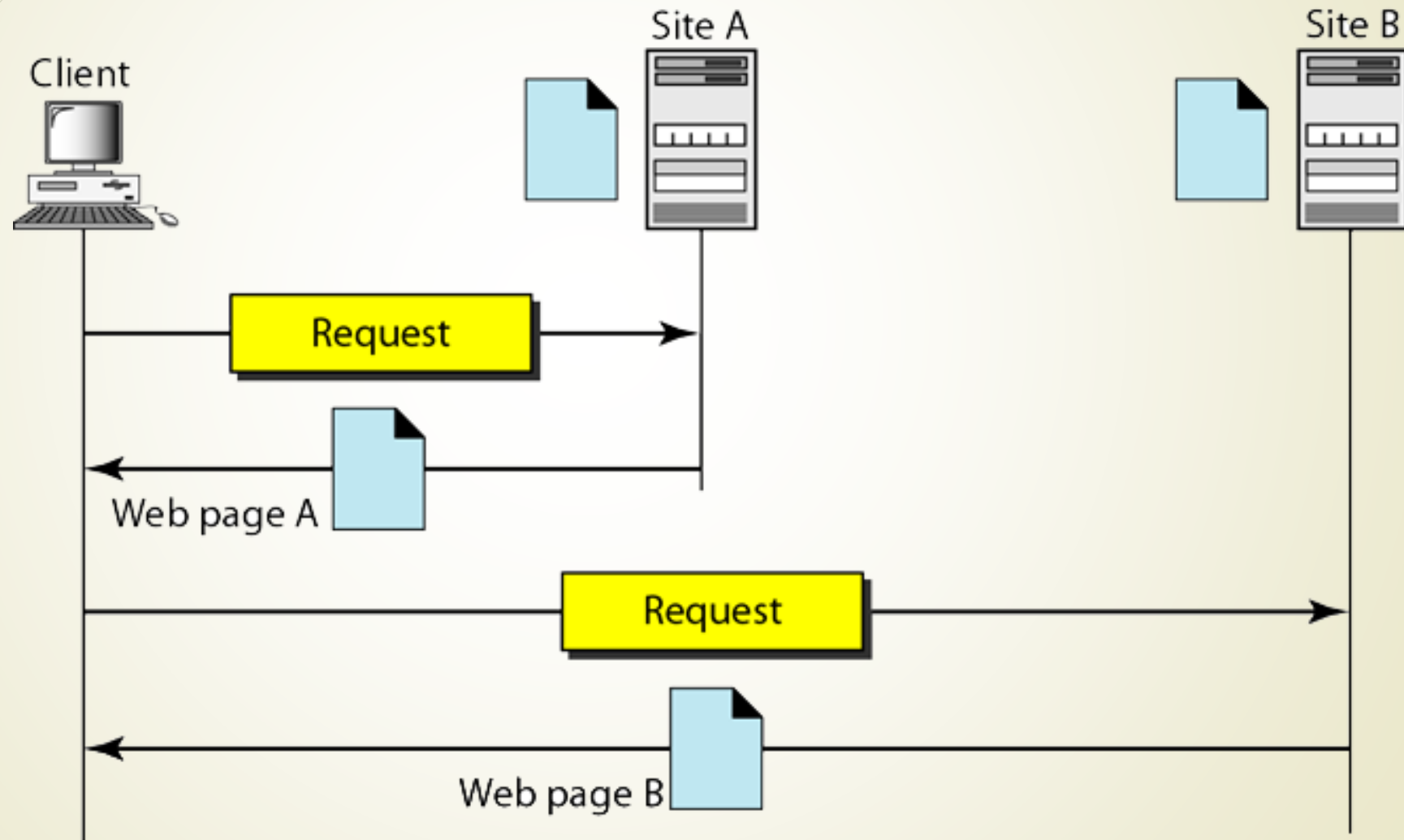


World Wide Web

What is WWW?

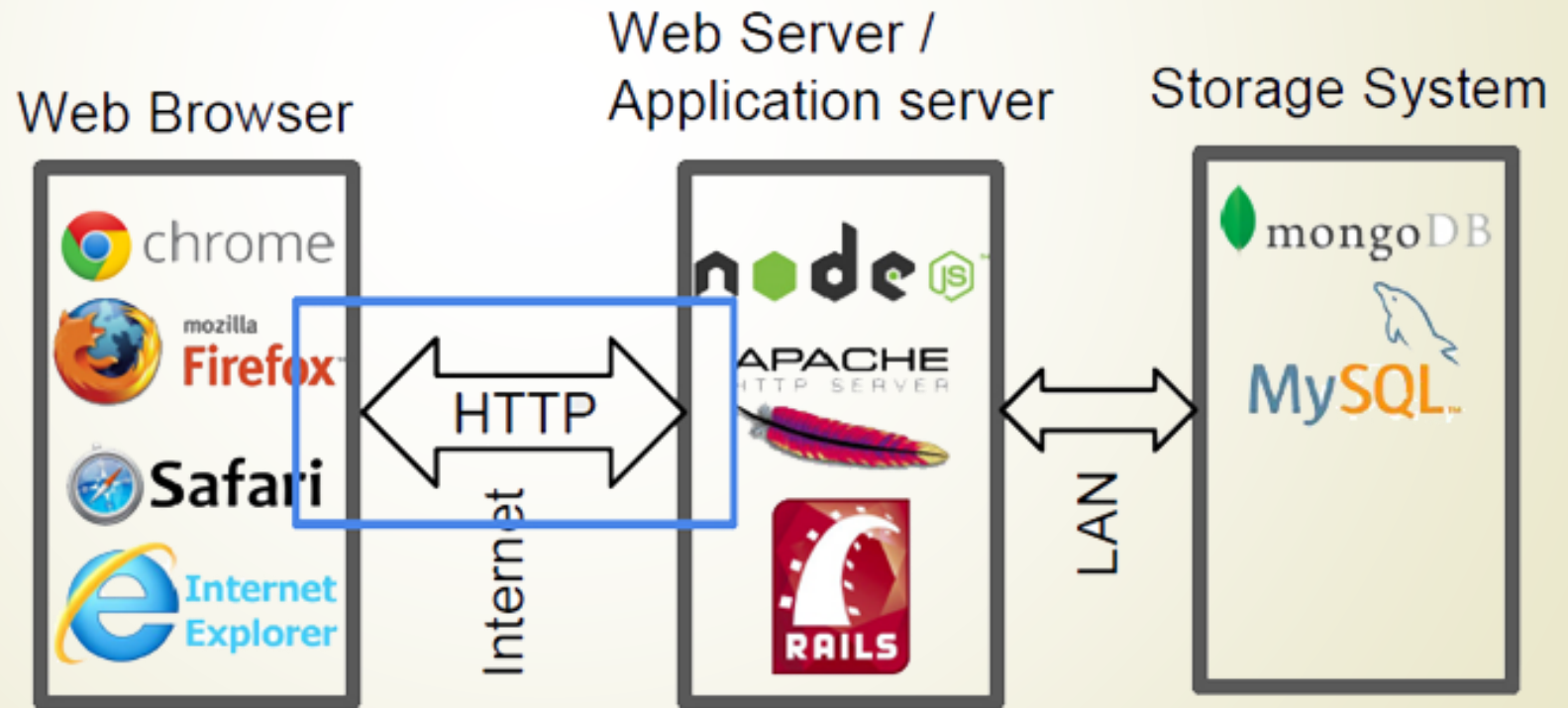
- ▶ **WWW = World Wide Web = Web != Internet**
- ▶ Internet is a global system of interconnected computer networks
- ▶ WWW is one of the services running in these networks
 - ▶ Global distributed information system in Internet (like E-mail, DNS, ...)

Architecture of WWW (1)



Architecture of WWW (2)

Web Application Architecture





WWW Components



- ▶ WWW structural components
 - ▶ **Internet** – provides data transfer channels over the TCP and HTTP
 - ▶ **Web servers** (Apache, IIS, Tomcat, Nginx, GWS, etc.) – serve HTTP
 - ▶ **Clients (Web browsers)** – download and display HTTP resources
- ▶ WWW semantic components
 - ▶ Hyper Text Transfer Protocol (**HTTP**) and other protocols
 - ▶ Uniform Resource Locator (**URL**), Uniform Resource Identifiers (**URI**)
 - ▶ Hyper Text Markup Language (**HTML**), Cascading Stylesheets (**CSS**)


Uniform Resource Locator (URL) (1)

- ▶ URL is an acronym that stands for *Uniform Resource Locator* and is a reference (an address) to a resource on the Internet.





What is a protocol?

- ▶ In diplomatic circles, a protocol is the set of rules governing a conversation between people
 - ▶ We have seen that the client and server carry on a machine-to-machine conversation
 - ▶ A network protocol is the set of rules governing a conversation between a client and a server
 - ▶ There are many protocols (BGP, DHCP, DNS, FTP, HTTP, IMAP, POP, RIP, SMTP, SNMP, SSH, Telnet, TLS/SSL, XMPP), HTTP is just one
- 

TCP/IP Layers & Protocols

HTTP sits top of the TCP/IP Protocol Stack

Application Layer

HTTP

Transport Layer

TCP

Network Layer

IP

Data Link Layer

Network Interfaces



What is a HTTP?

- HTTP = **H**ypertext **T**ransfer **P**rotocol
- The HTTP is an application layer protocol that allows web-based applications to communicate and exchange data.
- The HTTP is the Messenger of the web.
- It is a TCP/IP based protocol
- It used to deliver contents, for example, images, videos, audios, documents, etc.
- The computers that communicate via the HTTP must speak the HTTP protocol.
- HTTP is a stateless protocol.



History of HTTP Protocol

- ▶ HTTP/0.9 was the first one HTTP protocol and very simple, which only got one method, namely GET. The method could request a page from Server and the response always was an HTML page.
- ▶ HTTP/1.0 worked as the extension of HTTP/0.9. It expanded the protocol extended operations, extended negotiation, richer meta-information, tied with a security protocol and got more efficient by adding additional methods and header fields. But it uses a separate connection to the same server for every request-response transaction.
- ▶ While HTTP/1.1 can reuse a connection multiple times, for example, to download images or documents for a just delivered page. Hence HTTP/1.1 communications experience less latency as the establishment of TCP connections presents considerable overhead.



HTTP 0.9

```
$> telnet google.com 80
```

```
Connected to 74.125.xxx.xxx
```

```
GET /about/
```

```
(hypertext response)
```

```
(connection closed)
```

HTTP 1.0

```
$> telnet website.org 80

Connected to xxx.xxx.xxx.xxx

GET /rfc/rfc1945.txt HTTP/1.0 ①
User-Agent: CERN-LineMode/2.15 libwww/2.17b3
Accept: */*

HTTP/1.0 200 OK ②
Content-Type: text/plain
Content-Length: 137582
Expires: Thu, 01 Dec 1997 16:00:00 GMT
Last-Modified: Wed, 1 May 1996 12:45:26 GMT
Server: Apache 0.84

(plain-text response)
(connection closed)
```

- ① Request line with HTTP version number, followed by request headers
- ② Response status, followed by response headers

HTTP 1.1 (1)

```
$> telnet website.org 80
Connected to xxx.xxx.xxx.xxx

GET /index.html HTTP/1.1 ①
Host: website.org
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_7_4)... (snip)
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Encoding: gzip,deflate,sdch
Accept-Language: en-US,en;q=0.8
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.3
Cookie: __qca=P0-800083390... (snip)
```

```
HTTP/1.1 200 OK ②
Server: nginx/1.0.11
Connection: keep-alive
Content-Type: text/html; charset=utf-8
Via: HTTP/1.1 GWA
Date: Wed, 25 Jul 2012 20:23:35 GMT
Expires: Wed, 25 Jul 2012 20:23:35 GMT
Cache-Control: max-age=0, no-cache
Transfer-Encoding: chunked
```

```
100 ③
<!doctype html>
(snip)
```

```
100
(snip)
```

```
0 ④
```

① Request for HTML file, with encoding, charset, and cookie metadata

② Chunked response for original HTML request

③ Number of octets in the chunk expressed as an ASCII hexadecimal number (256 bytes)

④ End of chunked stream response

HTTP 1.1 (2)

0 4

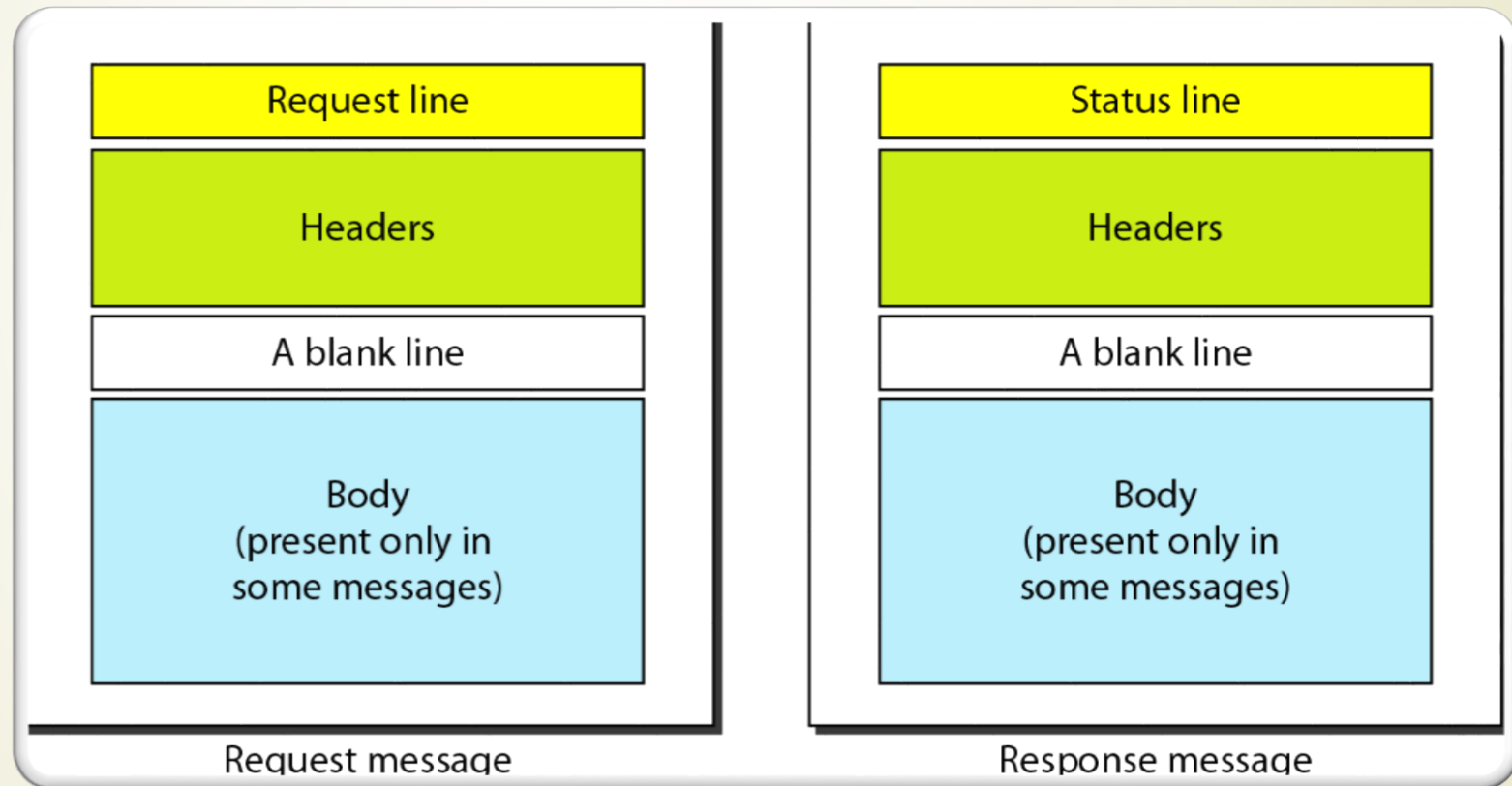
```
GET /favicon.ico HTTP/1.1 5
Host: www.website.org
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_7_4)... (snip)
Accept: */*
Referer: http://website.org/
Connection: close 6
Accept-Encoding: gzip,deflate,sdch
Accept-Language: en-US,en;q=0.8
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.3
Cookie: __qca=P0-800083390... (snip)
```

```
HTTP/1.1 200 OK 7
Server: nginx/1.0.11
Content-Type: image/x-icon
Content-Length: 3638
Connection: close
Last-Modified: Thu, 19 Jul 2012 17:51:44 GMT
Cache-Control: max-age=315360000
Accept-Ranges: bytes
Via: HTTP/1.1 GWA
Date: Sat, 21 Jul 2012 21:35:22 GMT
Expires: Thu, 31 Dec 2037 23:55:55 GMT
Etag: W/PSA-GAu26oXbDi
```

```
(icon data)
(connection closed)
```

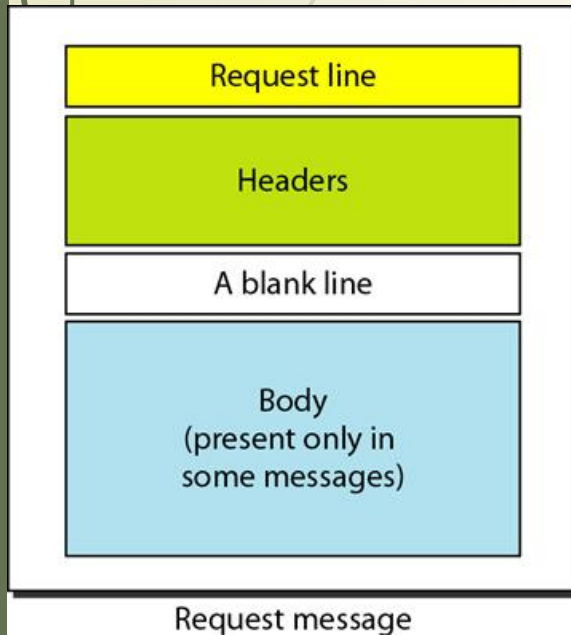
- 5 Request for an icon file made on same TCP connection
- 6 Inform server that the connection will not be reused
- 7 Icon response, followed by connection close

Request and Response Messages



HTTP Request

➤ Request message sent by a client consists of:



➤ HTTP request line (For example: GET www.yildiz.edu.tr HTTP/1.1)

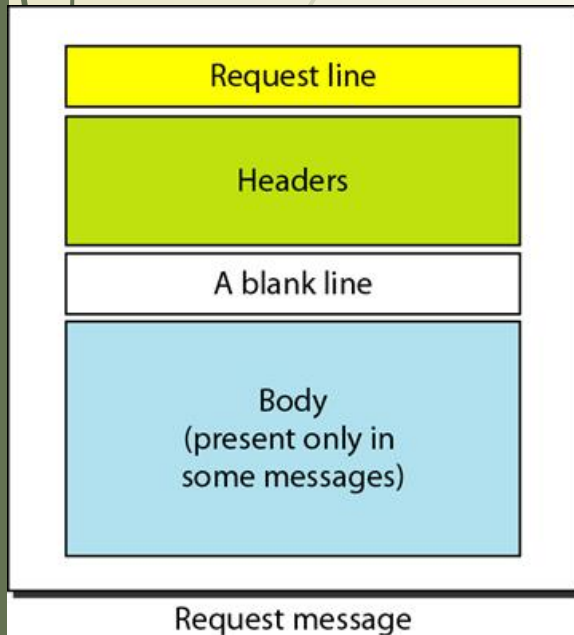
- Request method (GET / POST / PUT / DELETE / ...)
- Resource URI (URL)
- Protocol version

➤ HTTP request headers

- Additional parameters

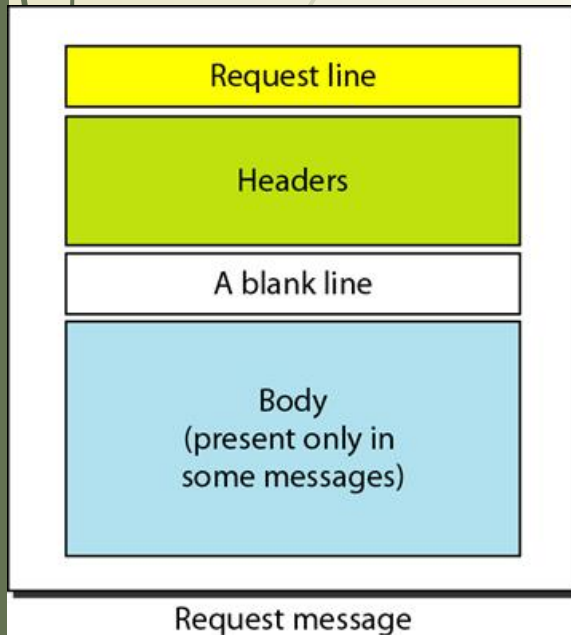
➤ HTTP body – optional data, e.g. posted form fields

Request Methods



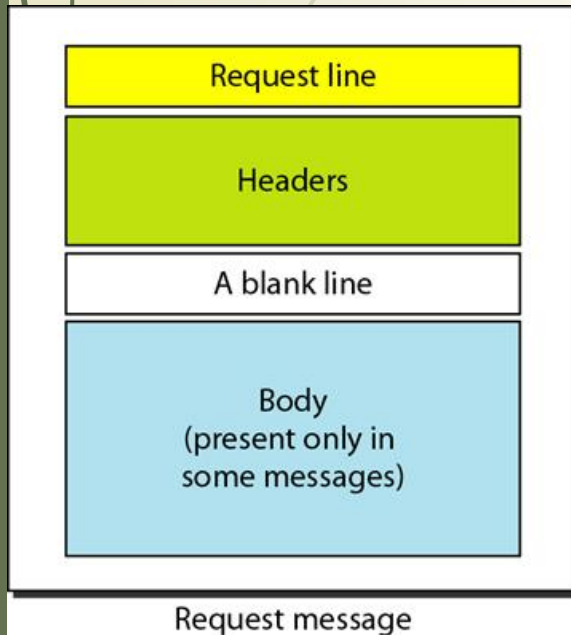
Command	Description
GET	Request for the resource located at the specified URL.
HEAD	It is like GET, but without the body of response.
POST	Submits data to be processed to the identified resource. The data is included in the body of the request.
PUT	Uploads a representation of the specified resource.
DELETE	Deletes the resource located at the specified URL.
TRACE	Echoes back the received request, so that a client can see what (if any) changes or additions have been made by intermediate servers.
OPTIONS	Returns the HTTP methods that the server supports for specified URL. This can be used to check the functionality of a web server by requesting '*' instead of a specific resource.
CONNECT	Converts the request connection to a transparent TCP/IP tunnel, usually to facilitate SSL-encrypted communication (HTTPS) through an unencrypted HTTP proxy.
PATCH	Is used to apply partial modifications to a resource.

Request Headers (1)



Header name	Description
Accept	Type of content accepted by the browser (for example text/html).
Accept-Charset	Character set expected by the browser
Accept-Encoding	Data coding accepted by the browser
Accept-Language	Language expected by the browser (English by default)
Authorization	Identification of the browser to the server

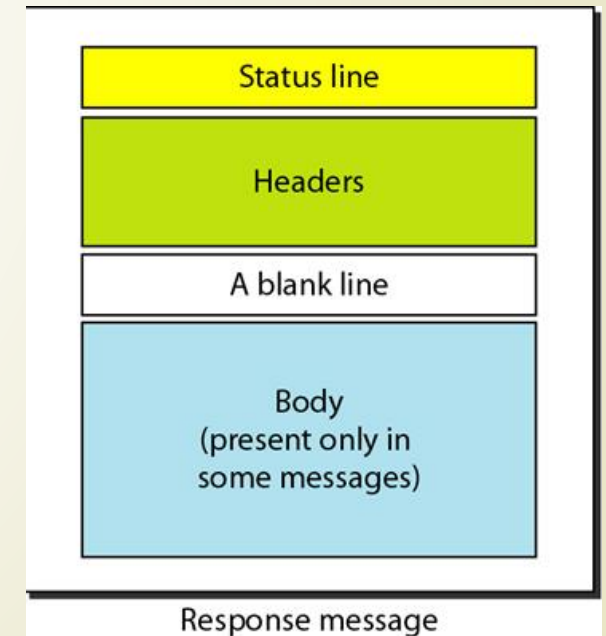
Request Headers (2)



Header name	Description
Host	The hostname (and optionally port) of server to which request is being sent
Referer	The URL of the resource from which the current request URI came
User-Agent	Name of the requesting application, used in browser sensing
Cookie	How clients pass cookies back to the servers that set them
Connection	Control options for the current connection and list of hop-by-hop request fields.

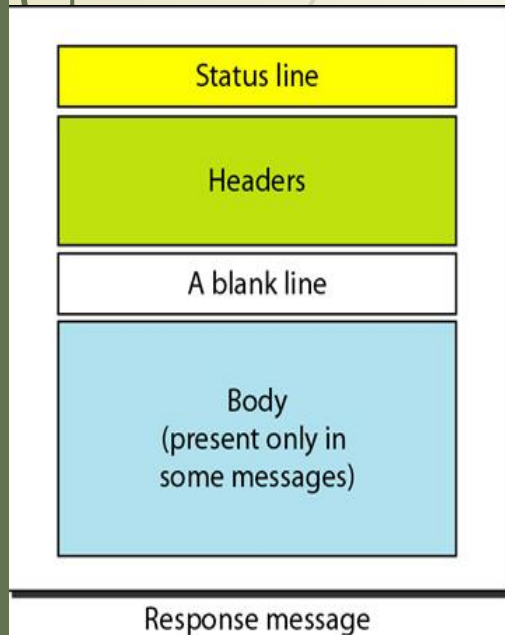
HTTP Response

- ▶ The response message sent by the HTTP server consists of:
 - ▶ HTTP response status line (For example: HTTP/1.1 200 OK)
 - ▶ Protocol version
 - ▶ Status code
 - ▶ Status phrase
 - ▶ Response headers
 - ▶ Provide meta data about the returned resource
 - ▶ Response body
 - ▶ The content of the HTTP response (data)



Response status code (1)

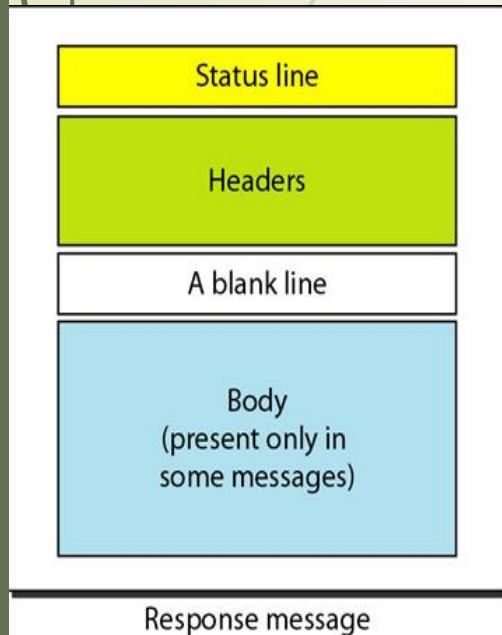
1xx: Information



Message:	Description:
100 Continue	The server has received the request headers, and the client should proceed to send the request body
101 Switching Protocols	The requester has asked the server to switch protocols
103 Checkpoint	Used in the resumable requests proposal to resume aborted PUT or POST requests

Response status code (2)

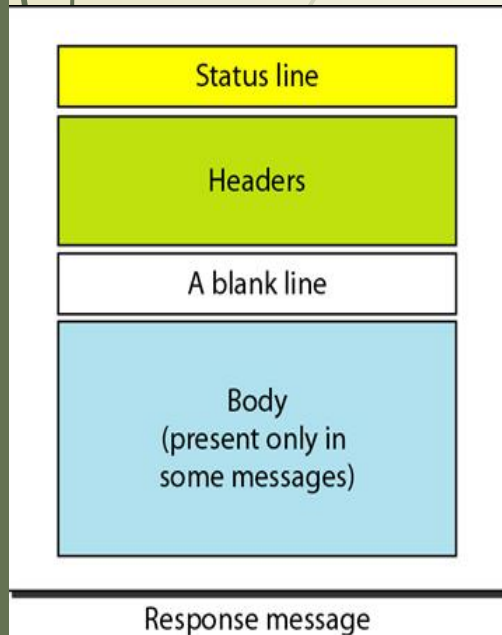
2xx: Successful



Message:	Description:
200 OK	The request is OK (this is the standard response for successful HTTP requests)
201 Created	The request has been fulfilled, and a new resource is created
202 Accepted	The request has been accepted for processing, but the processing has not been completed
203 Non-Authoritative Information	The request has been successfully processed, but is returning information that may be from another source
204 No Content	The request has been successfully processed, but is not returning any content
205 Reset Content	The request has been successfully processed, but is not returning any content, and requires that the requester reset the document view
206 Partial Content	The server is delivering only part of the resource due to a range header sent by the client

Response status code (3)

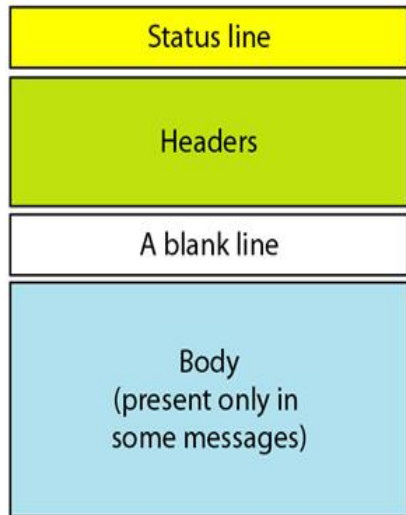
3xx: Redirection



Message:	Description:
300 Multiple Choices	A link list. The user can select a link and go to that location. Maximum five addresses
301 Moved Permanently	The requested page has moved to a new URL
302 Found	The requested page has moved temporarily to a new URL
303 See Other	The requested page can be found under a different URL
304 Not Modified	Indicates the requested page has not been modified since last requested
306 Switch Proxy	No longer used
307 Temporary Redirect	The requested page has moved temporarily to a new URL
308 Resume Incomplete	Used in the resumable requests proposal to resume aborted PUT or POST requests

Response status code (4)

4xx: Client Error

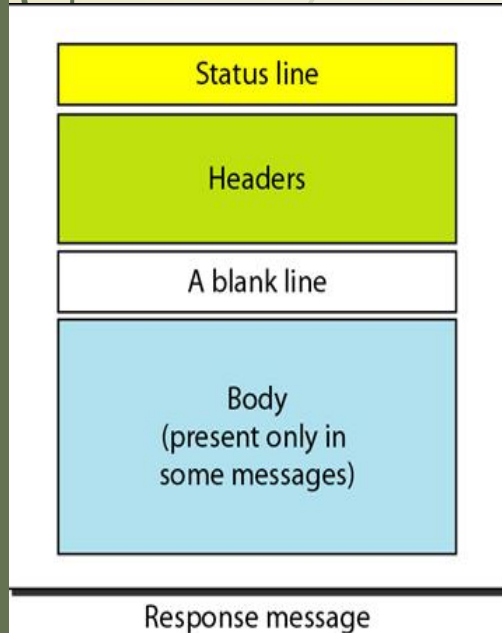


Response message

Message:	Description:
400 Bad Request	The request cannot be fulfilled due to bad syntax
401 Unauthorized	The request was a legal request, but the server is refusing to respond to it. For use when authentication is possible but has failed or not yet been provided
402 Payment Required	Reserved for future use
403 Forbidden	The request was a legal request, but the server is refusing to respond to it
404 Not Found	The requested page could not be found but may be available again in the future
405 Method Not Allowed	A request was made of a page using a request method not supported by that page
406 Not Acceptable	The server can only generate a response that is not accepted by the client
407 Proxy Authentication Required	The client must first authenticate itself with the proxy
408 Request Timeout	The server timed out waiting for the request
409 Conflict	The request could not be completed because of a conflict in the request
410 Gone	The requested page is no longer available
411 Length Required	The "Content-Length" is not defined. The server will not accept the request without it
412 Precondition Failed	The precondition given in the request evaluated to false by the server
413 Request Entity Too Large	The server will not accept the request, because the request entity is too large
414 Request-URI Too Long	The server will not accept the request, because the URL is too long. Occurs when you convert a POST request to a GET request with a long query information
415 Unsupported Media Type	The server will not accept the request, because the media type is not supported
416 Requested Range Not Satisfiable	The client has asked for a portion of the file, but the server cannot supply that portion
417 Expectation Failed	The server cannot meet the requirements of the Expect request-header field

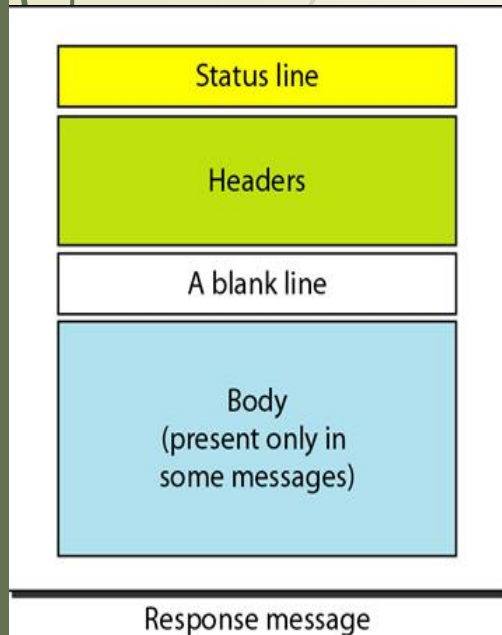
Response status code (5)

5xx: Server Error



Message:	Description:
500 Internal Server Error	A generic error message, given when no more specific message is suitable
501 Not Implemented	The server either does not recognize the request method, or it lacks the ability to fulfill the request
502 Bad Gateway	The server was acting as a gateway or proxy and received an invalid response from the upstream server
503 Service Unavailable	The server is currently unavailable (overloaded or down)
504 Gateway Timeout	The server was acting as a gateway or proxy and did not receive a timely response from the upstream server
505 HTTP Version Not Supported	The server does not support the HTTP protocol version used in the request
511 Network Authentication Required	The client needs to authenticate to gain network access

Response Header



Header name	Description
Content-Encoding	Type of coding for the body of the response
Content-Language	Type of language in the body of the response
Content-Length	Length of the body of the response
Content-Type	Type of content of the body of the response (for example text/html).
Date	Date data transfer starts.
Expires	Data use by date
Forwarded	Used by intermediary machines between the browser and server
Location	Redirection to a new URL associated with the document
Server	Features of the server having sent the response
Set-Cookie	This is how a server sets a cookie on a client



Entity Header

- ▶ Request and Response messages MAY transfer an entity if not otherwise restricted by the request method or response status code.

entity-header = Allow

- | Content-Encoding
- | Content-Language
- | Content-Length
- | Content-Location
- | Content-MD5
- | Content-Range
- | Content-Type
- | Expires
- | Last-Modified



General Headers

- ▶ General headers are used primarily to communicate information about the message itself, as opposed to what content it carries.
- ▶ They provide general information and control how a message is processed and handled.

general-header = Cache-Control

| Connection

| Date

| Pragma

| Trailer

| Transfer-Encoding

| Upgrade

| Via

| Warning



HTTPS

=

HTTP + SSL



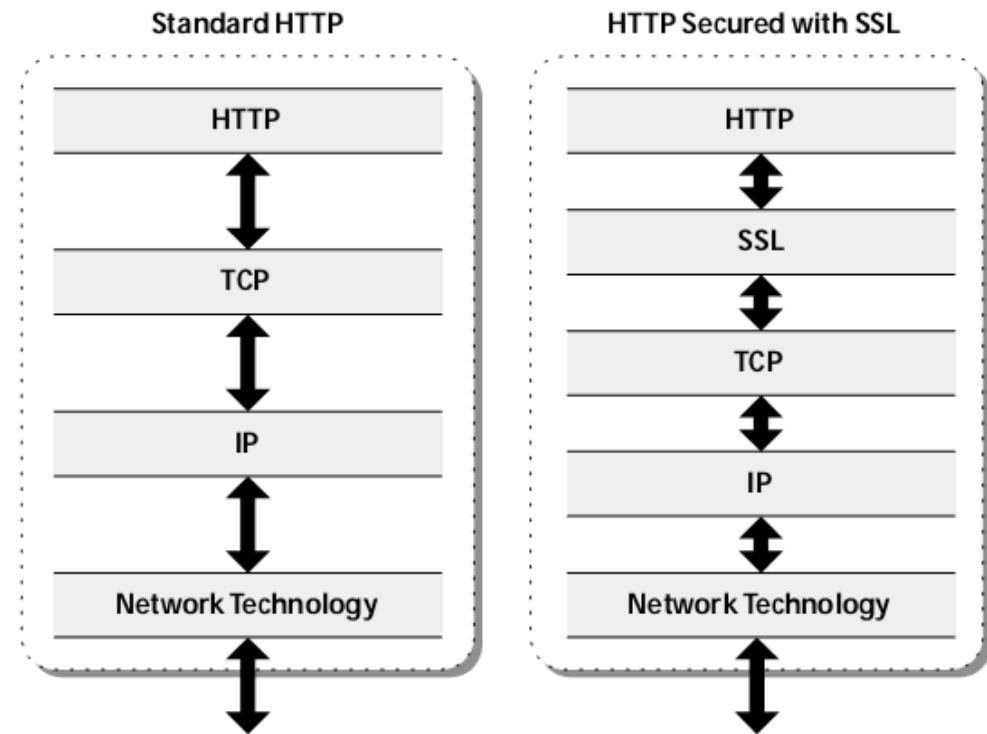
HTTPS (1)

- (HTTPS) Hypertext Transfer Protocol over Secure Socket Layer (SSL).
- First implementation of HTTP over SSL was issued in 1995 by Netscape.

HTTPS (2)

Figure 4.10 ►

The SSL protocol inserts itself between an application like HTTP and the TCP transport layer. TCP sees SSL as just another application, and HTTP communicates with SSL much the same as it does with TCP.



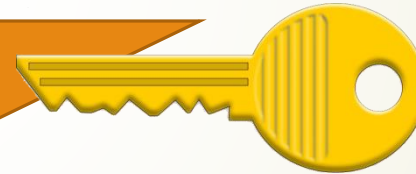
Cryptography (1)

Important information Data, Data, Data.

Encryption

*Encryption
Algorithm =
cipher*

Plain Text



Some random String

Hh2sh!~hH==E#@ns8676%===sdf

Cipher Text

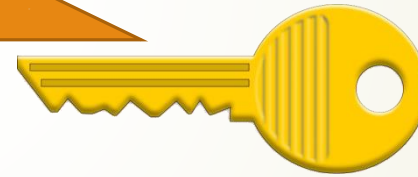
Cryptography (2)

Important information Data, Data, Data.



Symmetric Key

**Decryption
Algorithm**



Some random String

Hh2sh!~hH==E#@ns8676%===sdf

Asymmetric (public-key) encryption

Important information Data, Data, Data.

Encrypt



Public
Key

Hh2sh!~hH==E#@ns8676%===sdf

Decrypt



Private
Key

Important information Data, Data, Data.



SSL Session

- ▶ Uses asymmetric encryption to privately share the session key
 - ▶ Asymmetric has a lot of overhead
- ▶ Uses symmetric encryption to encrypt data
 - ▶ Symmetric encryption is quicker and uses less resource



ÖNEMLİ

Bu projeler lisansüstü öğrencilerinin hazırladığı çalışmalar olup tüm sorumluluk hazırlayan öğrencilere aittir. Öğrenciler hazırladığı projeye göre not almışlardır.