Application Layer protocols: HTTP

Application Layer Protocols

- Uses the services of transport layer protocols (TCP/UDP) to enable different application specific services to users
- Large number of protocols, too many to list
  - Telnet, FTP, SSH, HTTP, IMAP, POP, SMTP, RDP, RTP, ….
- We will cover basics of HTTP as an example

HTTP

- HyperText Transfer Protocol
- TCP based protocol to transfer objects between a HTTP client and server on the internet
- Standardized versions in use
  - HTTP 1.1
    - Supported by all websites
  - HTTP 2.0
    - Standardized in 2015, widely adopted
    - Around 50% of the websites support
- HTTP 3.0 also there, not standardized yet
- We will cover some basics of HTTP 1.1

Uniform Resource Locators (URL)

- Identifier for resources/objects on the Internet
- Three main components
  - Scheme: the protocol to be used to access the resource
    - http, https, ftp….
  - Host: identifies the host the resource is in
    - Can also be followed by a port
  - Path: identifies the resource in the host, or a subpart of the resource
- Example
  - http://www.abc.com/docs/paper1.pdf
  - ftp://www.abc.com:21/docs/paper1.pdf
  - http://www.abc.com/products/Model328#Specs
- Can also specify query strings, parameters to pass to scripts

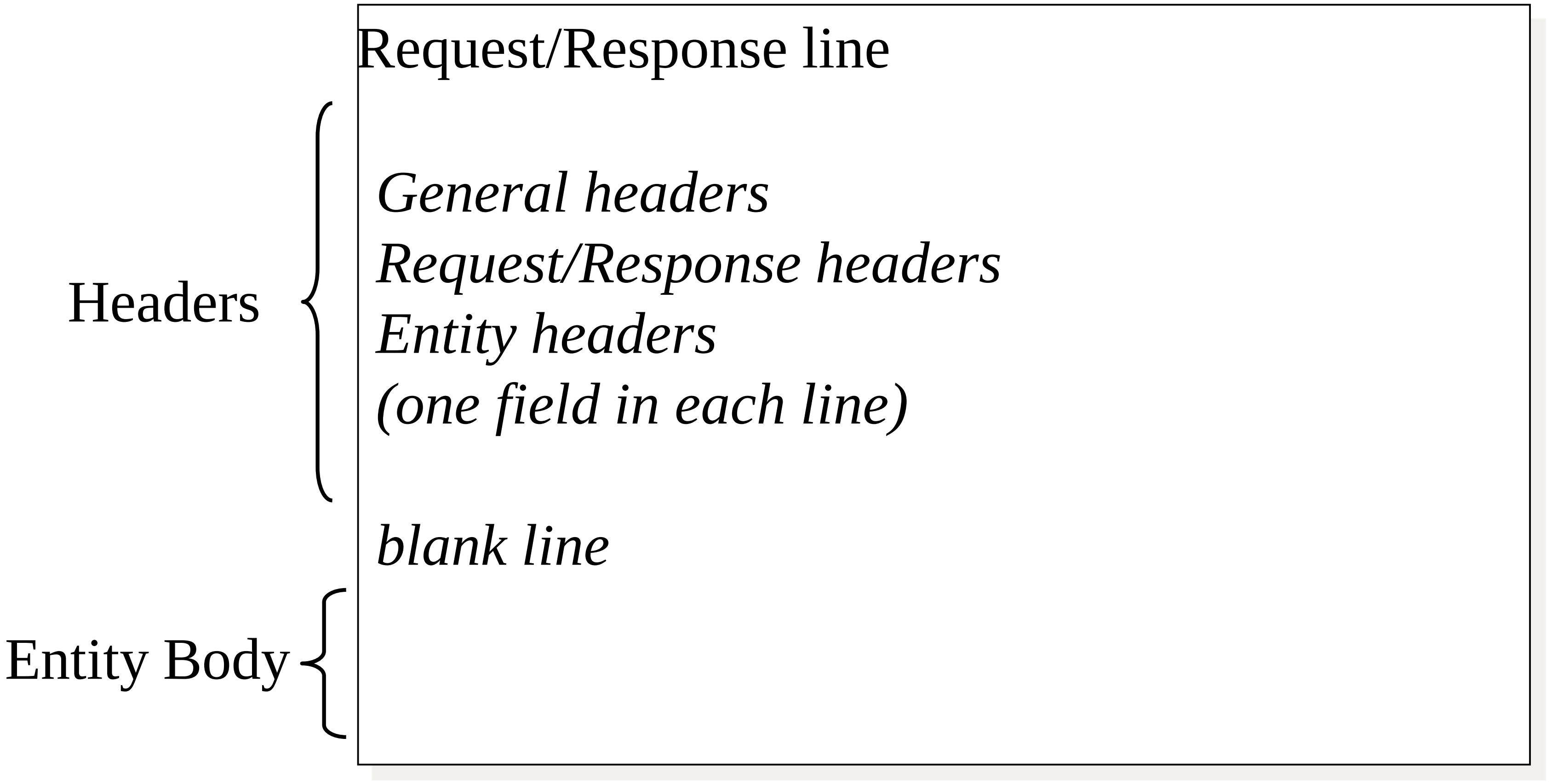
Basic HTTP Operation

- Client-server based operation
- Client opens TCP connection to server (default port 80 for http, port 443 for https)
- Client sends requests to server
  - Methods along with resource URI and parameters
- Server sends response
  - Always have status code to indicate success/error
  - May have content depending on request method
- Presentation of the content is not part of the protocol
- HTTP 1.1 is a text-based protocol, all requests and responses are sent as text
- HTTP is stateless, server need not remember any client state

Methods

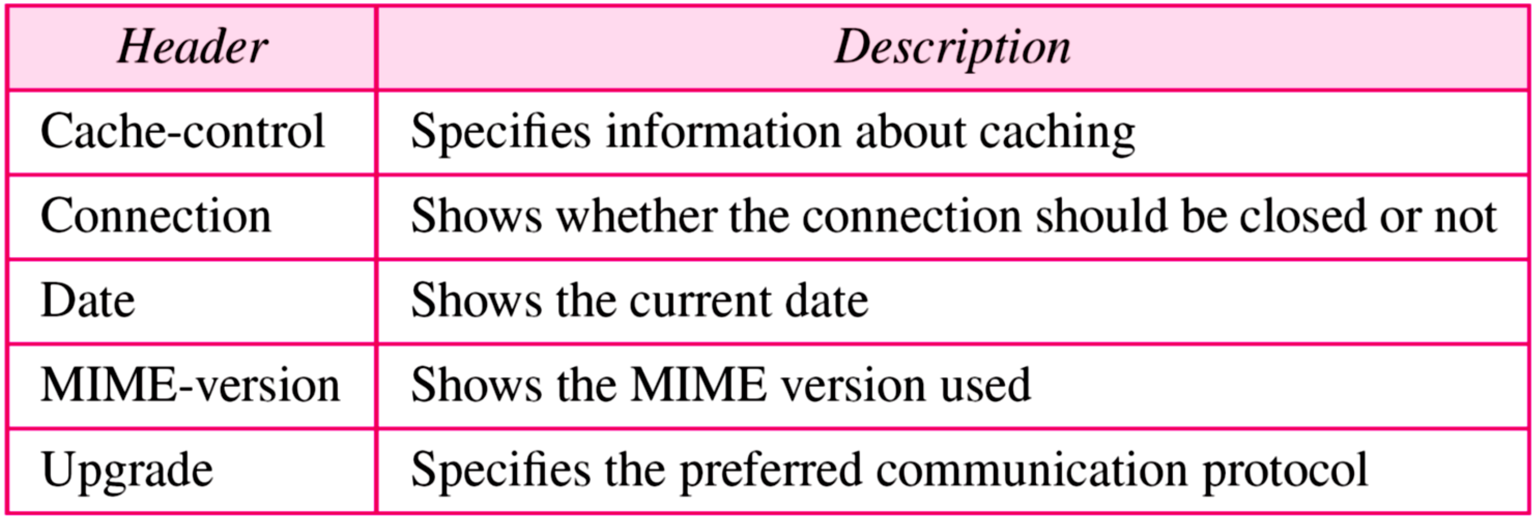
- GET: return the contents of a resource
  - Ex. A webpage, a file,…
- HEAD: return the header, without the actual contents
  - Useful for testing validity of the URL, or for collecting meta-data for the resource
- POST: Treat the document as a script and send some data to it
  - Ex. when forms are submitted
- PUT: Replace the contents of a resource with some data
- DELETE Delete the resource
- TRACE: Echo the incoming request
  - Useful for debugging
- OPTIONS: allows client to know what methods and headers can be used with a resource
- CONNECT: for connecting through proxies

HTTP Message Format



General headers

- Header fields that are applicable to both requests and response



Header	*Description*
Cache-control	Specifies information about caching
Connection	Shows whether the connection should be closed or not
Date	Shows the current date
MIME-version	Shows the MIME version used
Upgrade	Specifies the preferred communication protocol

Request Headers

- Header fields specific to request messages only

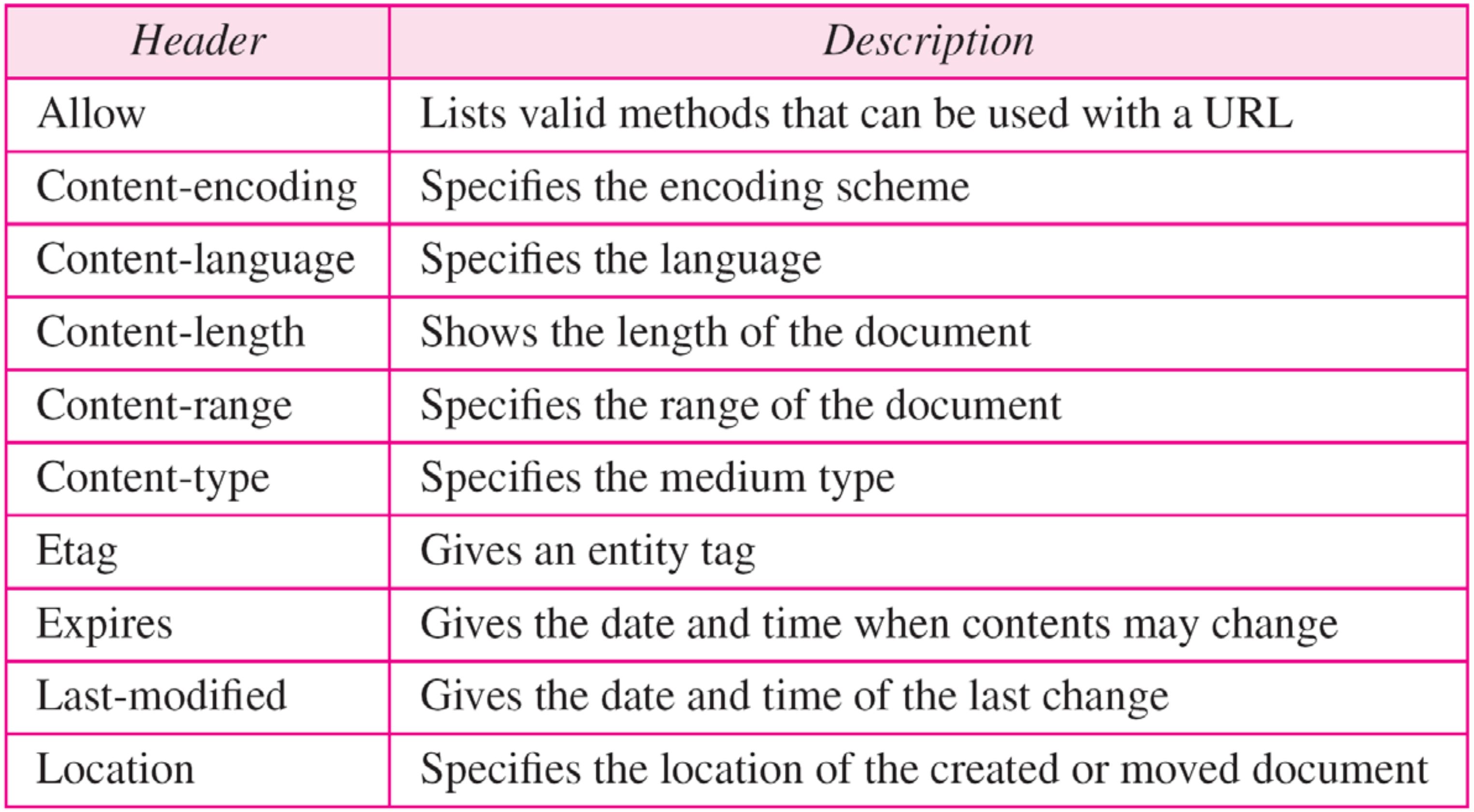
Header	*Description*
Accept	Shows the medium format the client can accept
Accept-charset	Shows the character set the client can handle
Accept-encoding	Shows the encoding scheme the client can handle
Accept-language	Shows the language the client can accept
Authorization	Shows what permissions the client has
From	Shows the e-mail address of the user
Host	Shows the host and port number of the server
If-modified-since	Sends the document if newer than specified date
If-match	Sends the document only if it matches given tag
If-non-match	Sends the document only if it does not match given tag
If-range	Sends only the portion of the document that is missing
If-unmodified-since	Sends the document if not changed since specified date
Referrer	Specifies the URL of the linked document
User-agent	Identifies the client program

HTTP Response Headers

Header	*Description*
Accept-range	Shows if server accepts the range requested by client
Age	Shows the age of the document
Public	Shows the supported list of methods
Retry-after	Specifies the date after which the server is available
Server	Shows the server name and version number

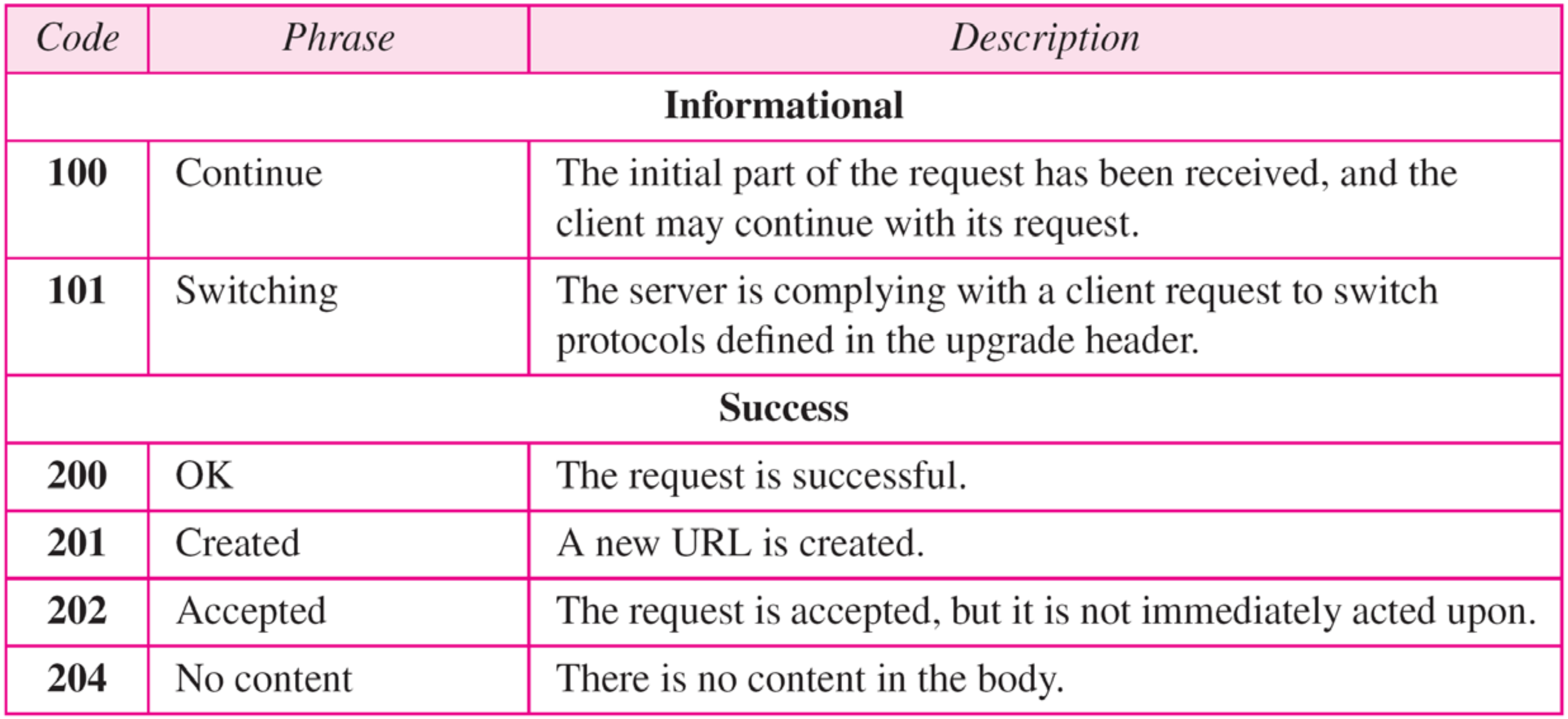
Entity Headers

- Header fields specifying attributes of the data sent



Header	*Description*
Allow	Lists valid methods that can be used with a URL
Content-encoding	Specifies the encoding scheme
Content-language	Specifies the language
Content-length	Shows the length of the document
Content-range	Specifies the range of the document
Content-type	Specifies the medium type
Etag	Gives an entity tag
Expires	Gives the date and time when contents may change
Last-modified	Gives the date and time of the last change
Location	Specifies the location of the created or moved document

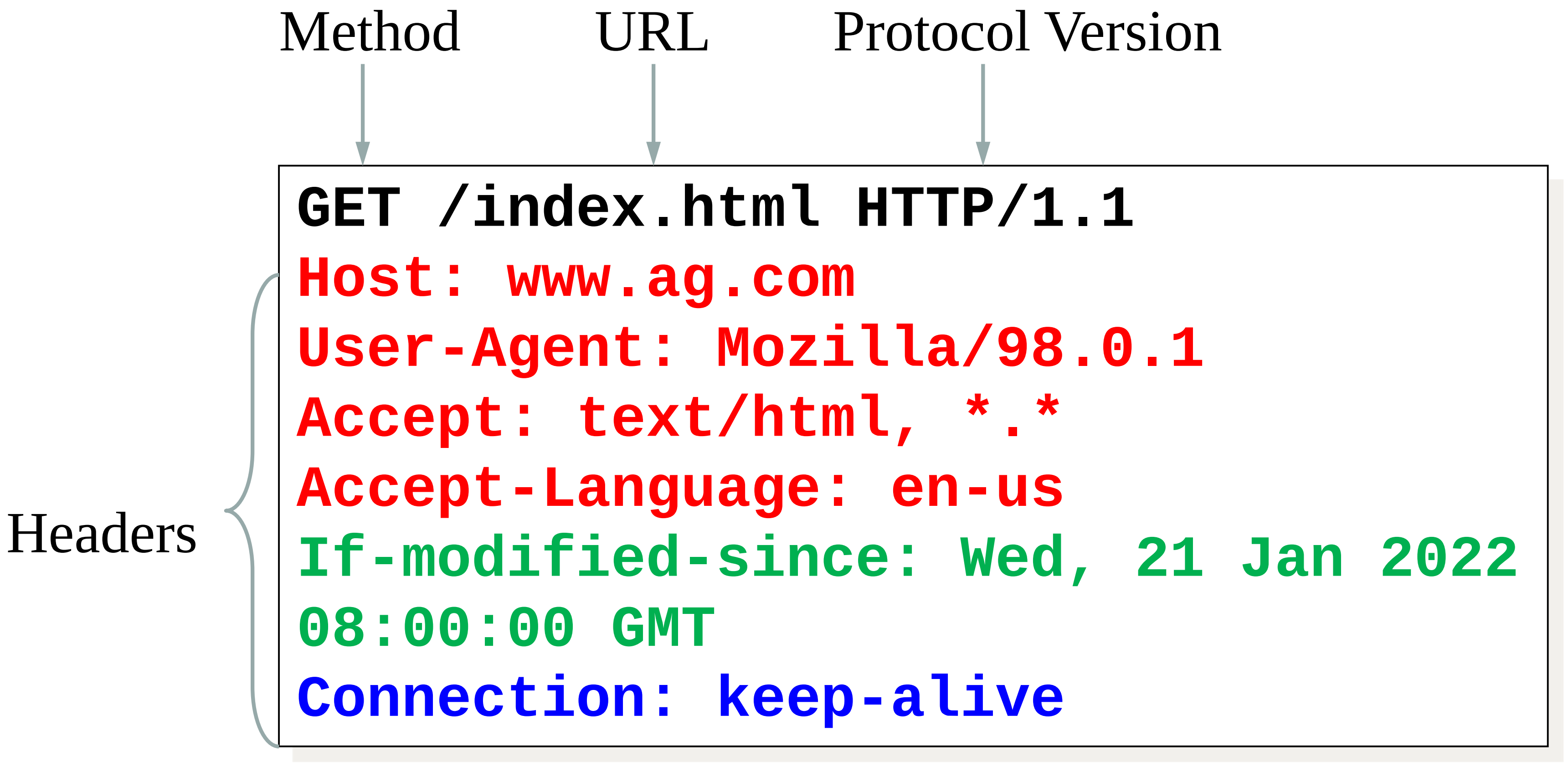
Status Codes



Code	*Phrase*	*Description*
		Informational
100	Continue	The initial part of the request has been received, and the client may continue with its request.
101	Switching	The server is complying with a client request to switch protocols defined in the upgrade header.
		Success
200	OK	The request is successful.
201	Created	A new URL is created.
202	Accepted	The request is accepted, but it is not immediately acted upon.
204	No content	There is no content in the body.

Code	*Phrase*	*Description*
	Redirection	
301	Moved permanently	The requested URL is no longer used by the server.
302	Moved temporarily	The requested URL has moved temporarily.
304	Not modified	The document has not been modified.
	Client Error	
400	Bad request	There is a syntax error in the request.
401	Unauthorized	The request lacks proper authorization.
403	Forbidden	Service is denied.
404	Not found	The document is not found.
405	Method not allowed	The method is not supported in this URL.
406	Not acceptable	The format requested is not acceptable.
	Server Error	
500	Internal server error	There is an error, such as a crash, at the server site.
501	Not implemented	The action requested cannot be performed.
503	Service unavailable	The service is temporarily unavailable, but may be requested in the future.

HTTP Request Example: GET



```
GET /index.html HTTP/1.1
Host: www.ag.com
User-Agent: Mozilla/98.0.1
Accept: text/html, *.*
Accept-Language: en-us
If-modified-since: Wed, 21 Jan 2022
08:00:00 GMT
Connection: keep-alive
```

General headers
Request headers
Entity headers

HTTP Response Example

Version Status Code Status Message

Headers

Entity Body

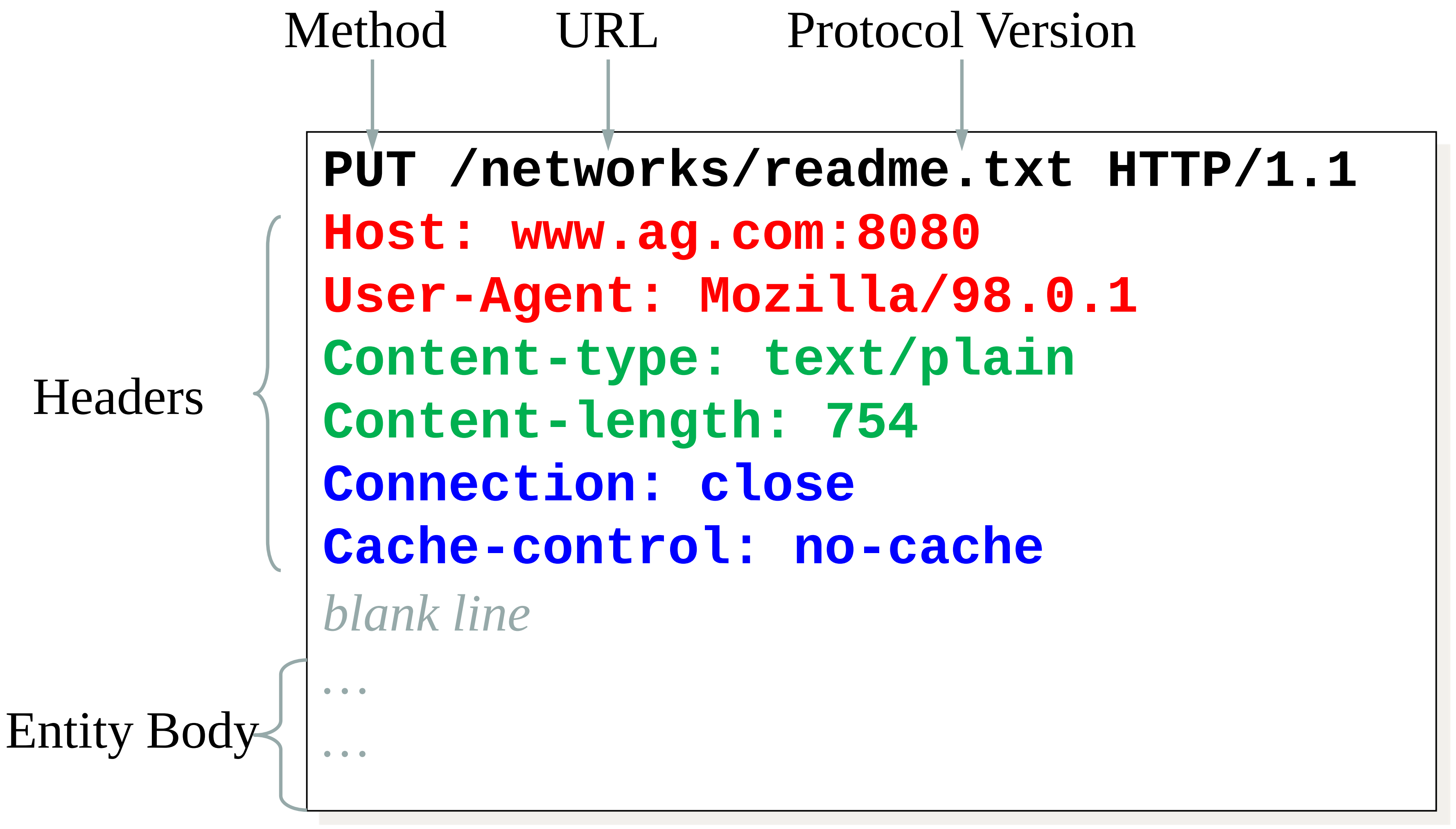
```
HTTP/1.1 200 OK
Date: Thu, 24 Mar 2022 17:43:21 GMT
Server: Apache/2.4.41
Content-Type: text/html
Content-Length: 1846
```

blank line

```
<html>
...
</html>
```

General headers
Response headers
Entity headers

HTTP Request Example: PUT



```
PUT /networks/readme.txt HTTP/1.1
Host: www.ag.com:8080
User-Agent: Mozilla/98.0.1
Content-type: text/plain
Content-length: 754
Connection: close
Cache-control: no-cache
```
blank line

. . .

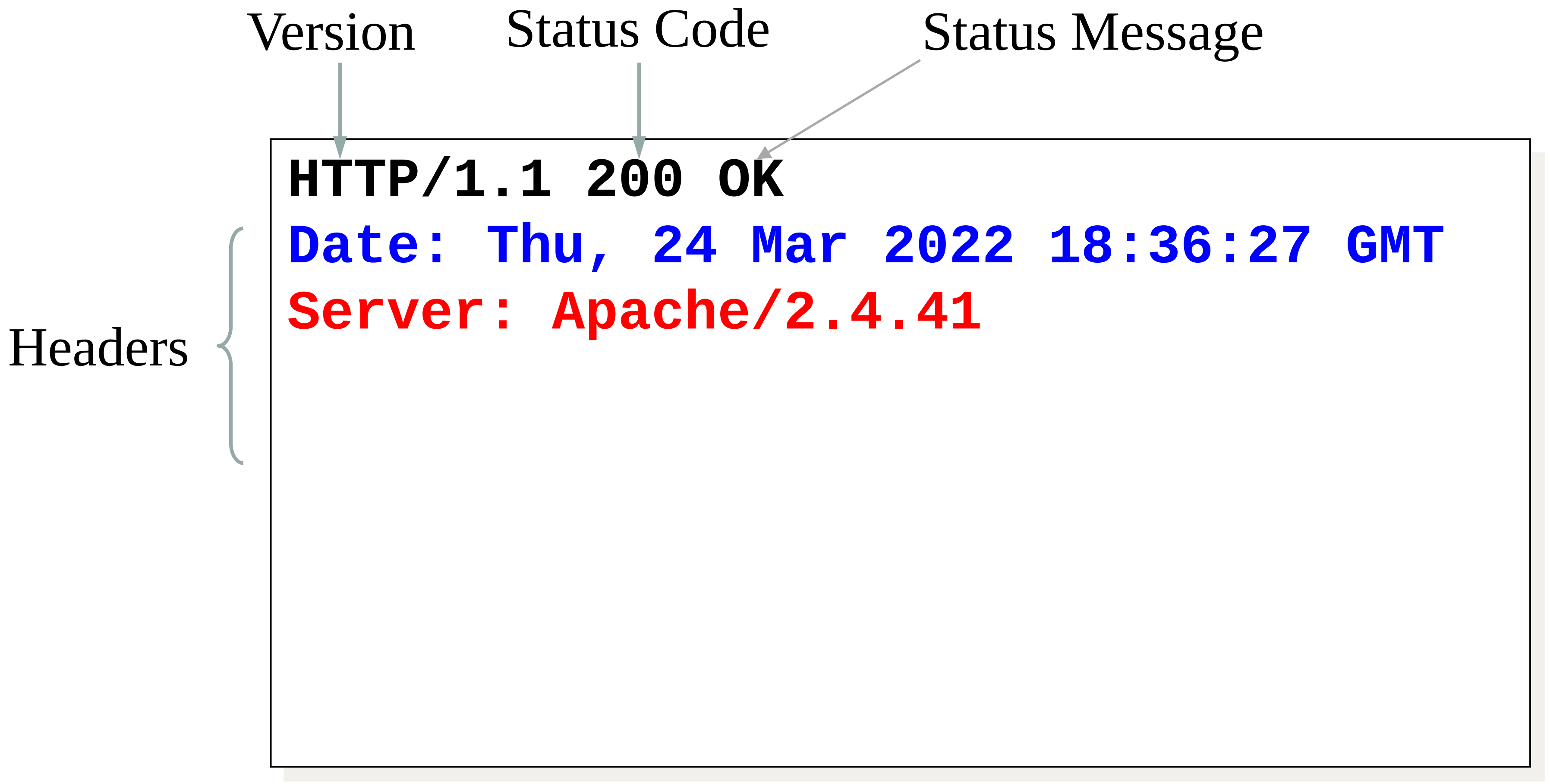
. . .

General headers

Response headers

Entity headers

HTTP Response Example



```
HTTP/1.1 200 OK
Date: Thu, 24 Mar 2022 18:36:27 GMT
Server: Apache/2.4.41
```

General headers
Response headers
Entity headers

Persistent Connection

- Non-persistent connection: one TCP connection made for each request-response
  - Inefficient when downloading say a page with lots of images etc. which will need to be downloaded anyway
- Persistent connection: A connection is reused for multiple request-response
  - Default for HTTP 1.1 onwards
  - The server can close the connection if client requests (using *connection* header field) or on timeout
- Persistent connections can also pipeline requests

Cookies

- Small pieces of data that the server stores on client side
  - Server sends cookies using *set-cookie* response header field
  - Multiple *set-cookie* fields can be used to send more than one piece of data

```
HTTP/2.0 200 OK
Content-Type: text/html
Set-Cookie: movie_seen=untouchables
Set-Cookie: actor=sean_connery
…
```

  - Can set expiry time on cookies also

```
Set-Cookie: name=ag; Expires=Thu, 30 APR 2022
10:30:00 GMT
```

- Client sends on subsequent requests to same server/domain
  - Uses the cookie field in request header

```
GET /arbit.html HTTP/1.1
Host: www.example.org
Cookie: movie_seen=untouchables;actor=sean_connery
```

- Allows servers to track users to push more personalized information like name, targeted advertisements, personalized movie recommendations etc.

HTTPS

- HTTP over a secure connection
- Uses Transport Layer Security (TLS)
  - Earlier known as Secure Socket layer (SSL)
- Allows for both client and server authentication using digital certificates
- Allows encrypted message exchange
- Security parameters (algorithms to use, keys etc.) negotiated at the start before any actual data transfer
- Interoperability maintained by negotiating capabilities of client and server at the beginning