The World Wide Web

Outline

- Background
- Structure
- Protocols

HTTP Basics

- Protocol for client/server communication
  - The heart of the Web
  - Very simple request/response protocol
    - Client sends request message, server replies with response message
  - Stateless
  - Relies on URI naming mechanism
- Three versions have been used
  - 09/1.0 – very close to Berners-Lee's original
    - RFC 1945 (original RFC is now expired)
  - 1.1 – developed to enhance performance, caching, compression
    - RFC 2068, standardized in 1997
  - 1.0 dominates
  - 2.0 Introduced in 2015
  - 3.0 coming up

HTTP Request Format

request-line (request method -URI HTTP-protocol version)
headers (0 or more)
<blank line>
body (only for POST request)

- First type of HTTP message: *requests*
  - Client browsers construct and send message
- Typical HTTP request:
  - GET http://www.cs.wisc.edu/index.html HTTP/1.0

Figure 22.10 *HTTP transaction*

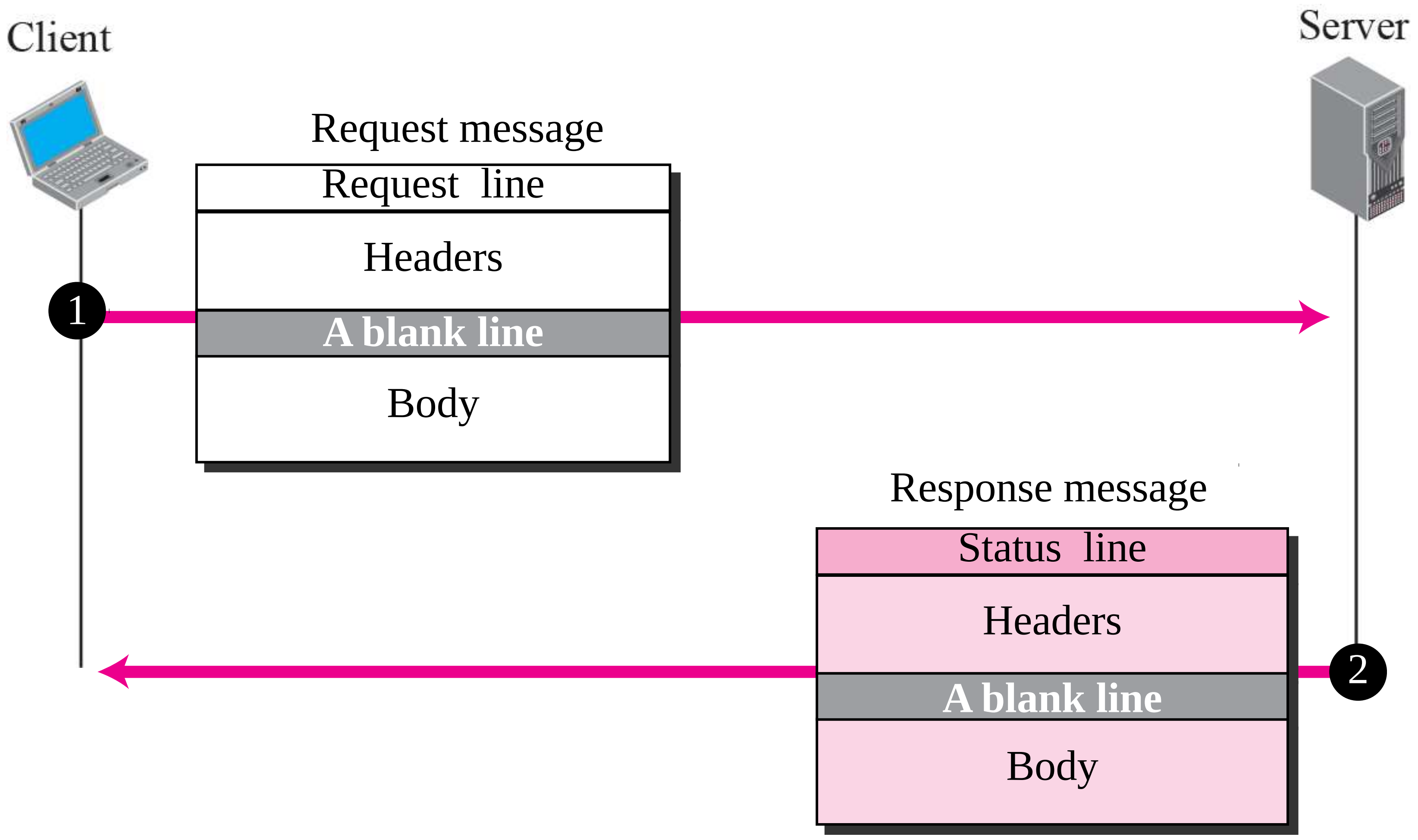
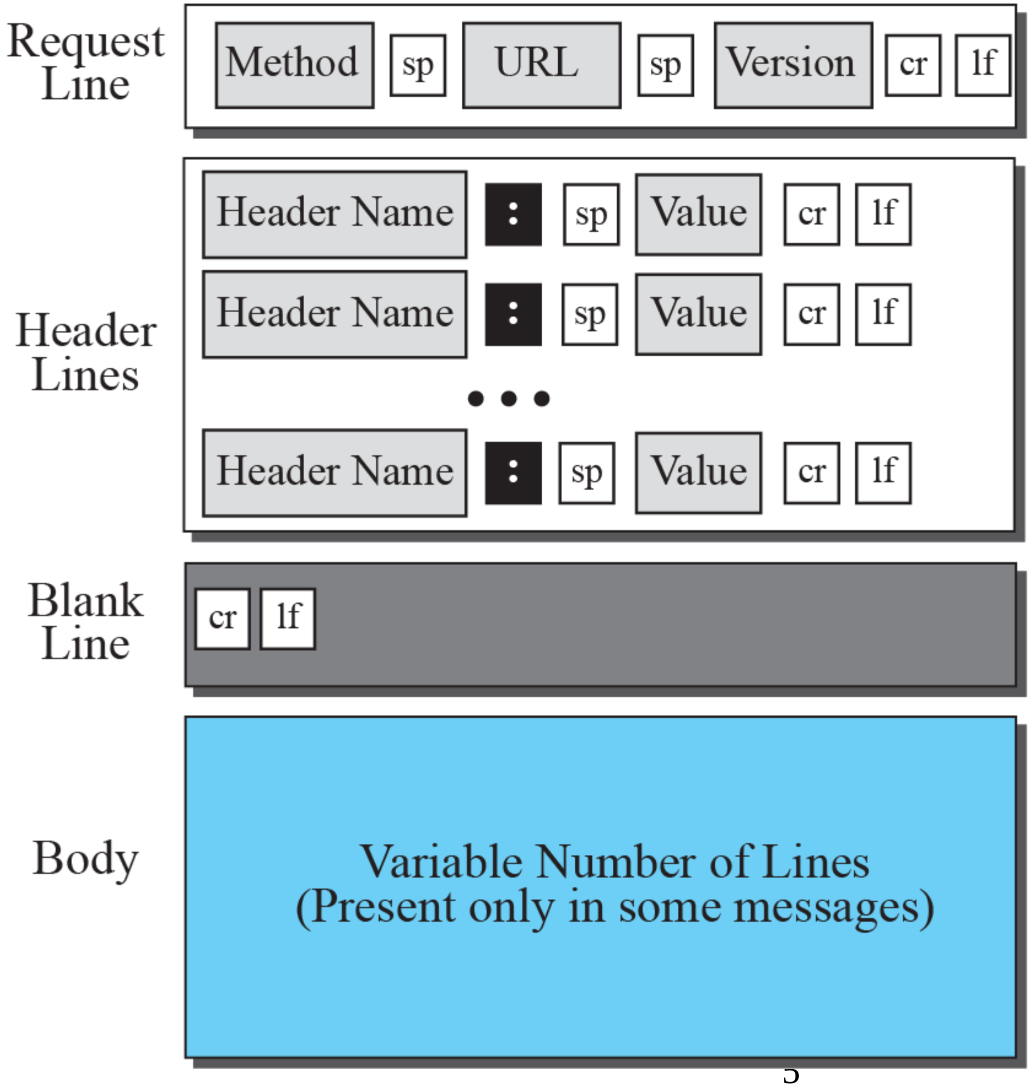


Figure 22.11 *Format of the request message*



Legend

sp: Space
cr: Carriage Return
lf: Line Feed

HTTP Request Methods

- GET – retrieve document specified by URL
- PUT – store specified document under given URL
- HEAD – retrieve info. about document specified by URL
- OPTIONS – retrieve information about available options
- POST – give information (eg. annotation) to the server
- DELETE – remove document specified by URL
- TRACE – loopback request message
- CONNECT – for use by caches

Table 22.2 *Request Header Names*

Header	*Description*
User-agent	Identifies the client program
Accept	Shows the media 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
Host	Shows the host and port number of the client
Date	Shows the current date
Upgrade	Specifies the preferred communication protocol
Cookie	Returns the cookie to the server
If-Modified-Since	Returns the cookie to the server

HTTP 1.0

- Other information
  - User Agent
    - identify the Operating System and Browser of the web-server.

    Click to add text

    User-Agent: Mozilla/4.0 (compatible; MSIE5.01; Windows NT)

  - If-Modified-Since
    - Returns object only if more recent than given date
    - Otherwise returns status code 304

HTTP 1.0 example

See word document

HTTP 1.0

- Other information
  - Accept
    - Mime types which browser can accept
      - Multipurpose Internet Mail Extension
        - » text/plain
        - » text/html
        - » application/postscript
        - » image/gif
        - » image/jpeg
        - » audio/basic
        - » video/mpeg
        - » x-world/x-vrml

HTTP Response Format

status-line (HTTP-version response-code response-phrase)
headers (0 or more)
<blank line>
body

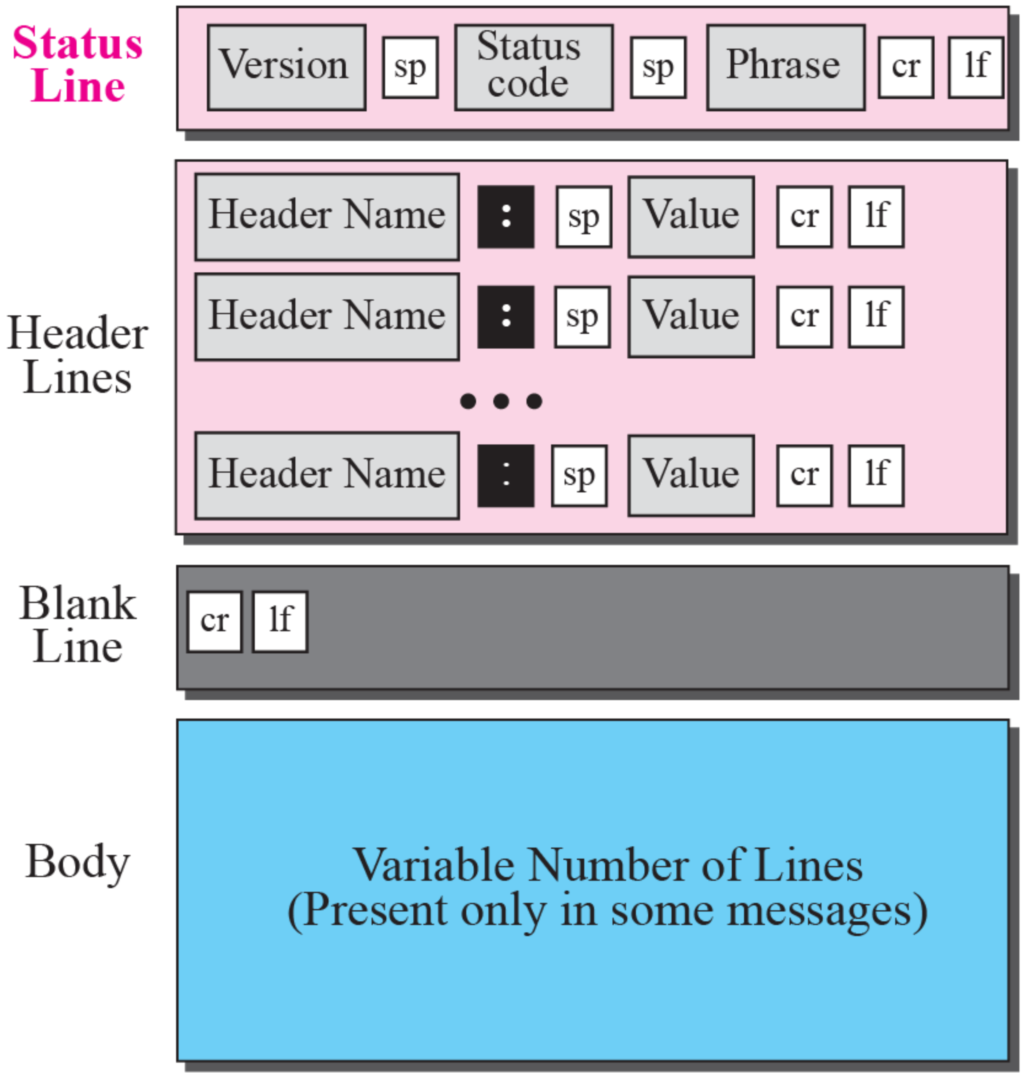
- Second type of HTTP message: *response*
  - Web servers construct and send response messages
- Typical HTTP response:
  - HTTP/1.0 301 Moved Permanently

    Location: http://www.wisc.edu/cs/index.html

HTTP Response Codes

- 1xx – Informational – request received, processing
- 2xx – Success – action received, understood, accepted
    - » 200 OK
    - » 201 POST command successful
    - » 202 Request accepted
    - » 203 GET or HEAD request fulfilled
    - » 204 No content
- 3xx – Redirection – further action necessary
  - – Further action must be taken in order to complete request
    - » 300 Resource found at multiple locations
    - » 301 Resource moved permanently
    - » 302 Resource moved temporarily
    - » 304 Resource has not modified (since date)

Figure 22.12 *Format of the response message*



Legend

sp: Space
cr: Carriage Return
lf: Line Feed

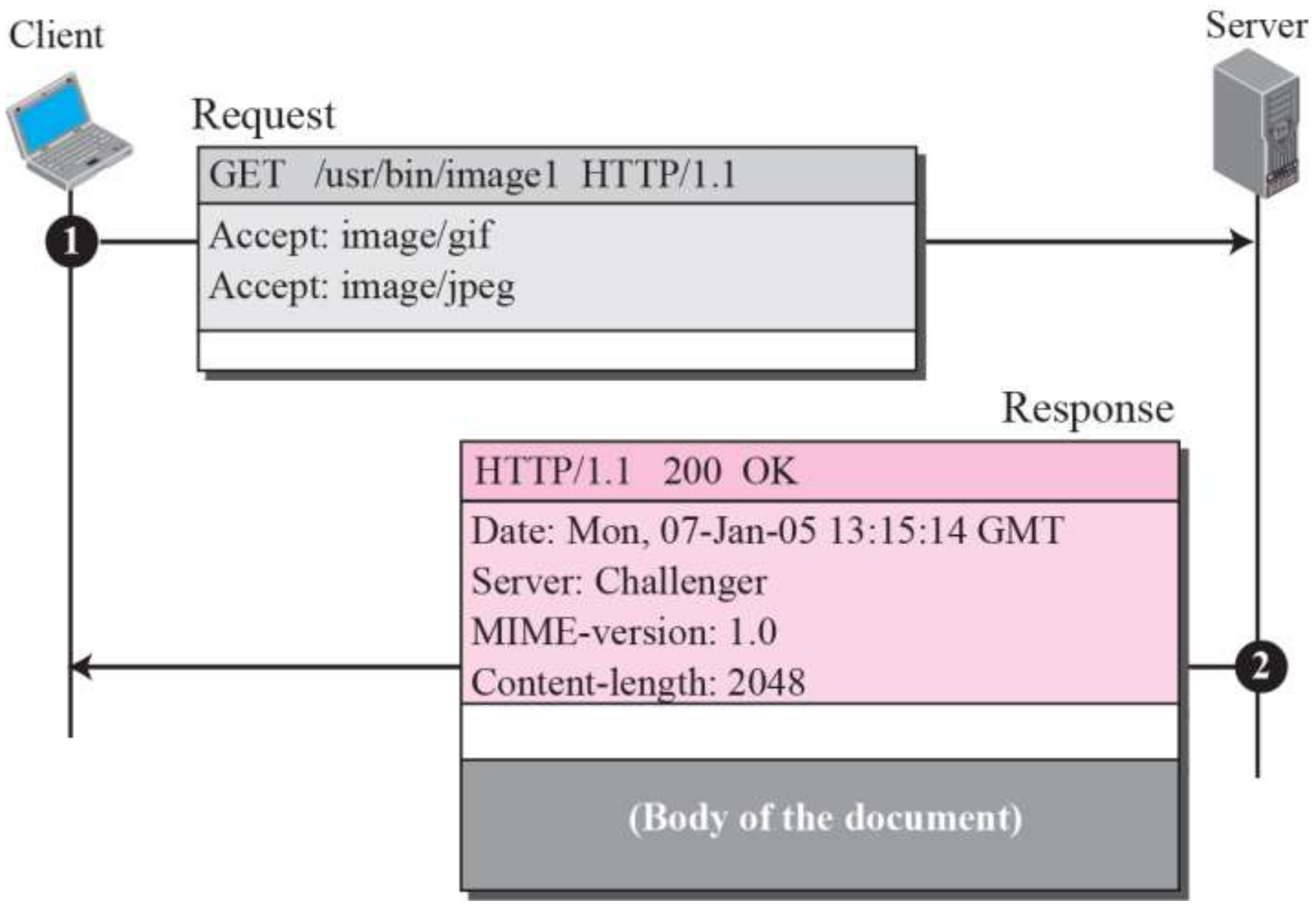
Table 22.3 *Status Codes and Status Phrases*

Status Code	*Status Phrase*	*Description*
	Informational	
100	Continue	The initial part of the request received, continue.
101	Switching	The server is complying to switch protocols.
	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.
	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 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.

Table 22.4 *Response Header Names*

Header	*Description*
Date	Shows the current date
Upgrade	Specifies the preferred communication protocol
Server	Gives information about the server
Set-Cookie	The server asks the client to save a cookie
Content-Encoding	Specifies the encoding scheme
Content-Language	Specifies the language
Content-Length	Shows the length of the document
Content-Type	Specifies the media type
Location	To ask the client to send the request to another site
Accept-Ranges	The server will accept the requested byte-ranges
Last-modified	Gives the date and time of the last change

Figure 22.13 *Example 22.4*



```
GET   /usr/bin/image1  HTTP/1.1
Accept: image/gif
Accept: image/jpeg
```

```
HTTP/1.1   200  OK
Date: Mon, 07-Jan-05 13:15:14 GMT
Server: Challenger
MIME-version: 1.0
Content-length: 2048
```

HTTP/1.0 Network Interaction

- Clients make requests to port 80 on servers
  - Uses DNS to resolve server name
- Clients make separate TCP connection for each URL
  - Some browsers open multiple TCP connections
    - Netscape default = 4
- Server returns HTML page
  - Many types of servers with a variety of implementations
  - Apache is the most widely used
    - Freely available in source form
- Client parses page
  - Requests embedded objects
- Problems
  - HTTP is stateless
    - Each request requires separate TCP connection
    - Server doesn’t remember previous requests

HTTP/1.0 Network Interaction

- Supports only GET, HEAD, POST

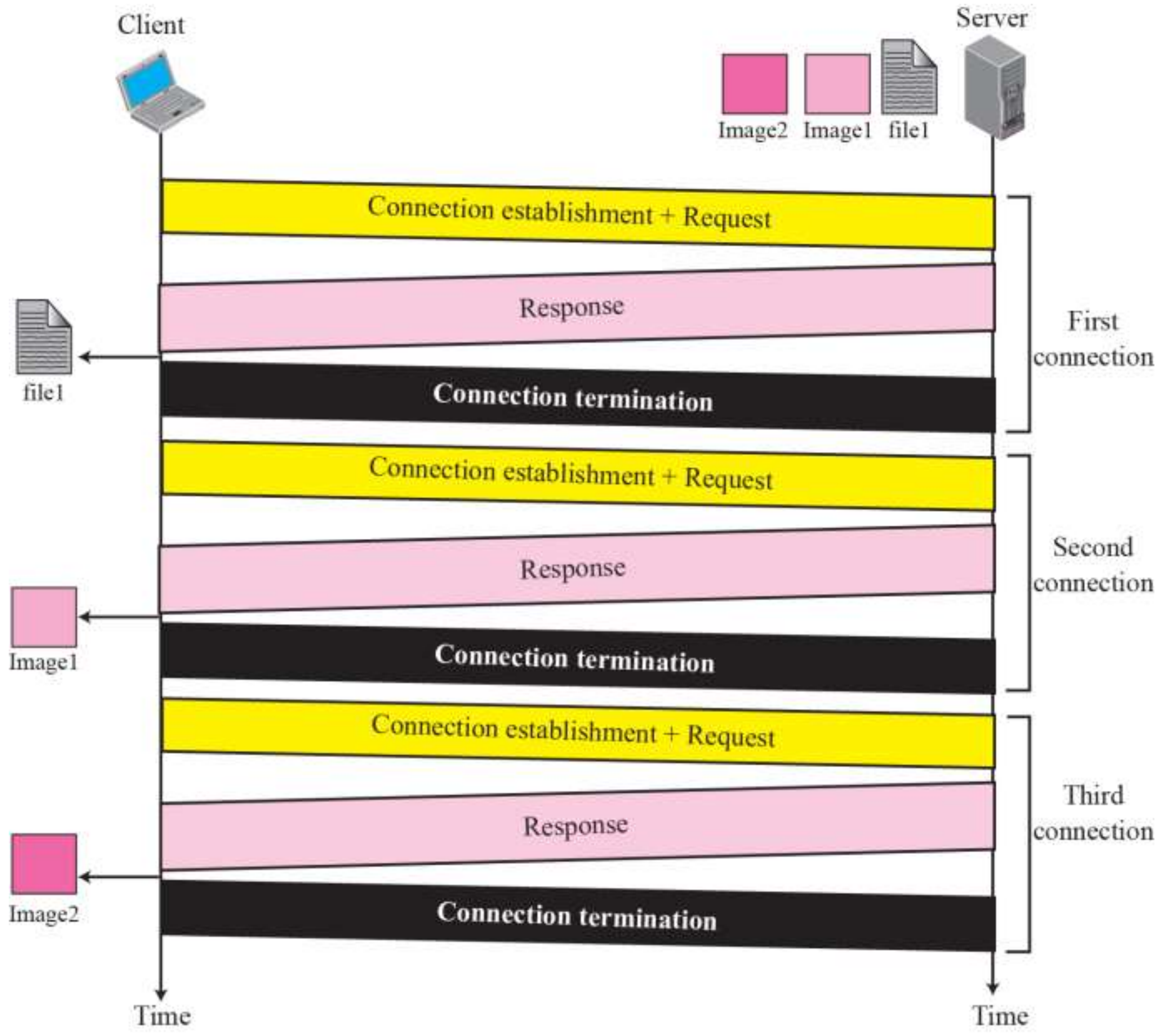
HTTP/1.1 Performance Enhancements

- HTTP/1.0 is a “stop and wait” protocol
  - Separate TCP connection for each file
    - Connect setup and tear down is incurred for each file
    - Inefficient use of packets
    - Server must maintain many connections in TIME_WAIT
- Mogul and Padmanabahn studied these issues in ’95
  - Resulted in HTTP/1.1 specification focused on performance enhancements
    - Persistent connections
    - Pipelining
    - Enhanced caching options
    - Support for compression

Evolution of HTTP

- HTTP/1.1
  - Persistent connections
    - In HTTP/1.0, if a single page includes inline images, multiple frames, animation, and other external references, to browse this page would require many reconnections
    - In HTTP/1.1 there are multiple request/response transactions per connection
    - Clients can pipeline requests to the server by sending multiple requests at start of session

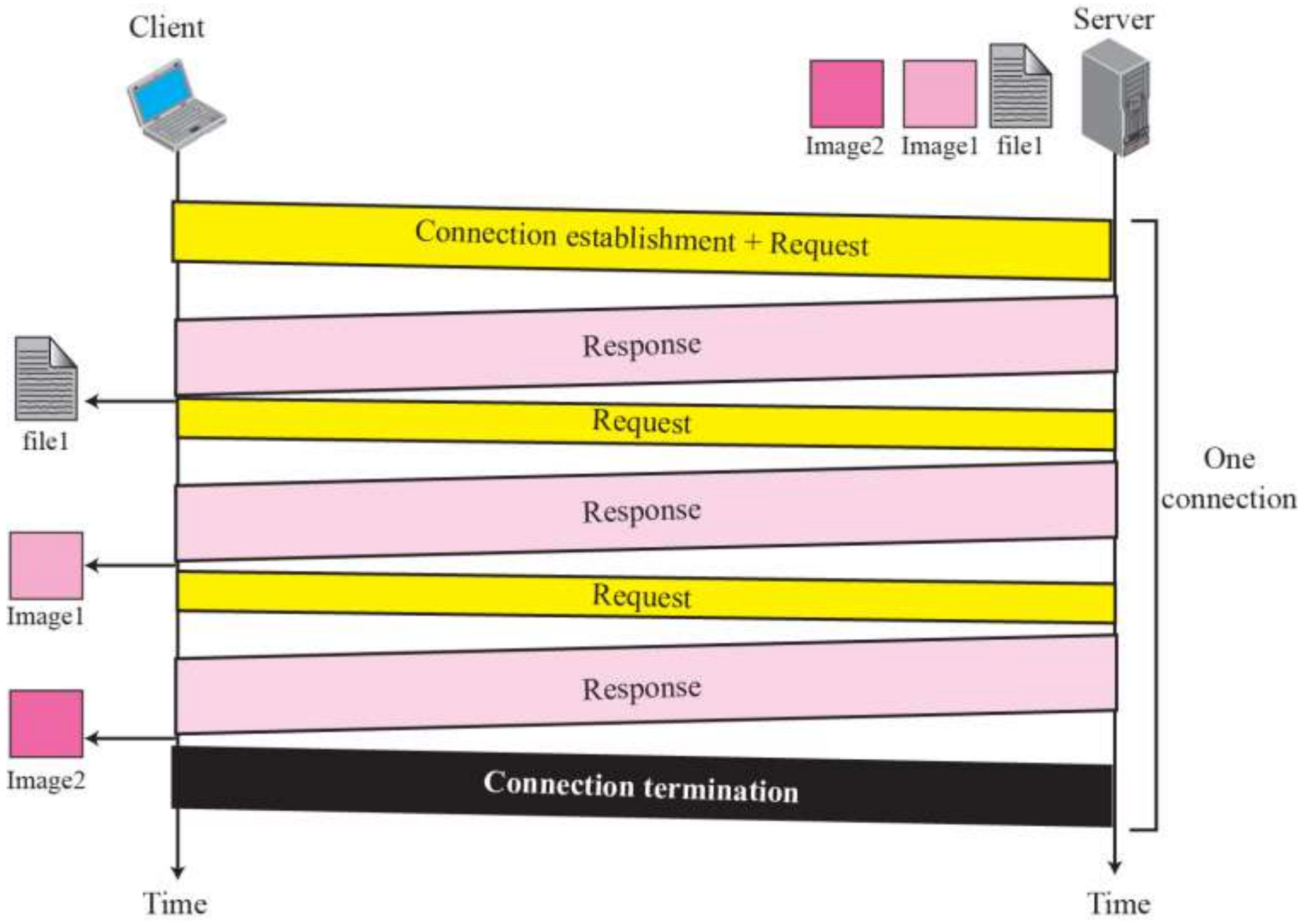
Figure 22.15 *Example 22.8*



Persistent Connections and Pipelining

- Persistent connections
  - Use the same TCP connection(s) for transfer of multiple files
  - Reduces packet traffic significantly
  - May or may not increase performance from client perspective
    - Load on server increases
- Pipelining
  - Pack as much data into a packet as possible
  - Requires length field(s) within header
  - May or may not reduce packet traffic or increase performance
    - Page structure is critical

Figure 22.16 *Example 22.9*



Evolution of HTTP

- HTTP/1.1
  - Cache management with entity tags
    - When body of URI changes, so does its entity tag
      - Useful for maintaining caches, as updated URI information would have a different entity tag
    - Can tell if same resource is being cached from multiple URI's as it would have same entity tag
    - Strong entity tag
      - Changes when any portion of resource changes
        - » One or more bytes change
    - Weak entity tag
      - Changes only when semantics of entity-body changes