EECS 489 - FA 24

Discussion 3

HTTP Protocol

Logistics

Assignment 1

Due: Monday, Sept. 18 @ 11:59 p.m.

- Only one weekend left!
- 3 Autograder submits per-day
- **There are no late days for this assignment!**

Autograder Submissions:

- Our AG was configured incorrectly earlier to reject files for Part 3 and Part 4 – **please resubmit your assignment with these included** ASAP if they were shown as discarded, even if you already got a 100.

Compute Usage: Make sure you turn off the instance before clicking “End Lab” to keep AWS costs manageable!

Discussions

- Aditya’s (4:30-5:30 Thurs) discussion will be recorded (and has been since Week 2).
  - You’re still encouraged to come in-person!
- All slides are available beforehand in the Google Drive.

HTTP Protocol

HTTP 1.1 Request

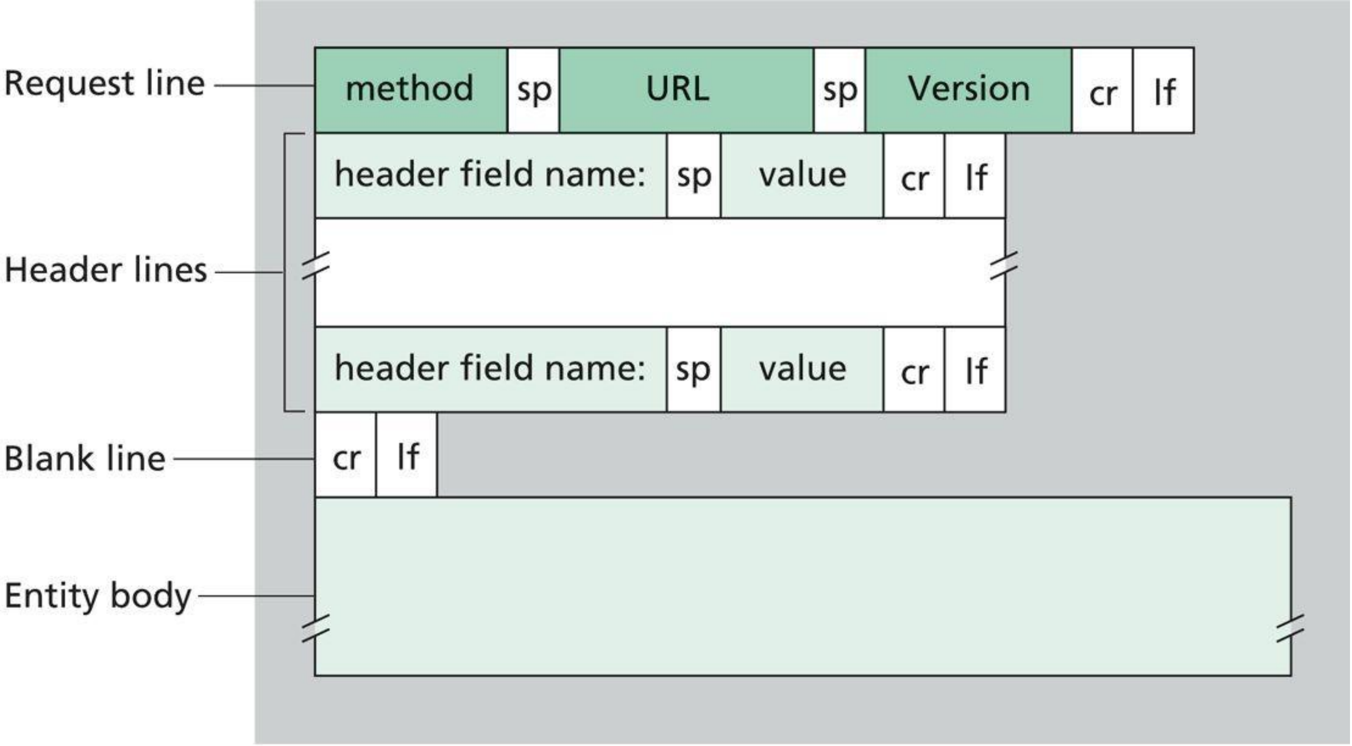
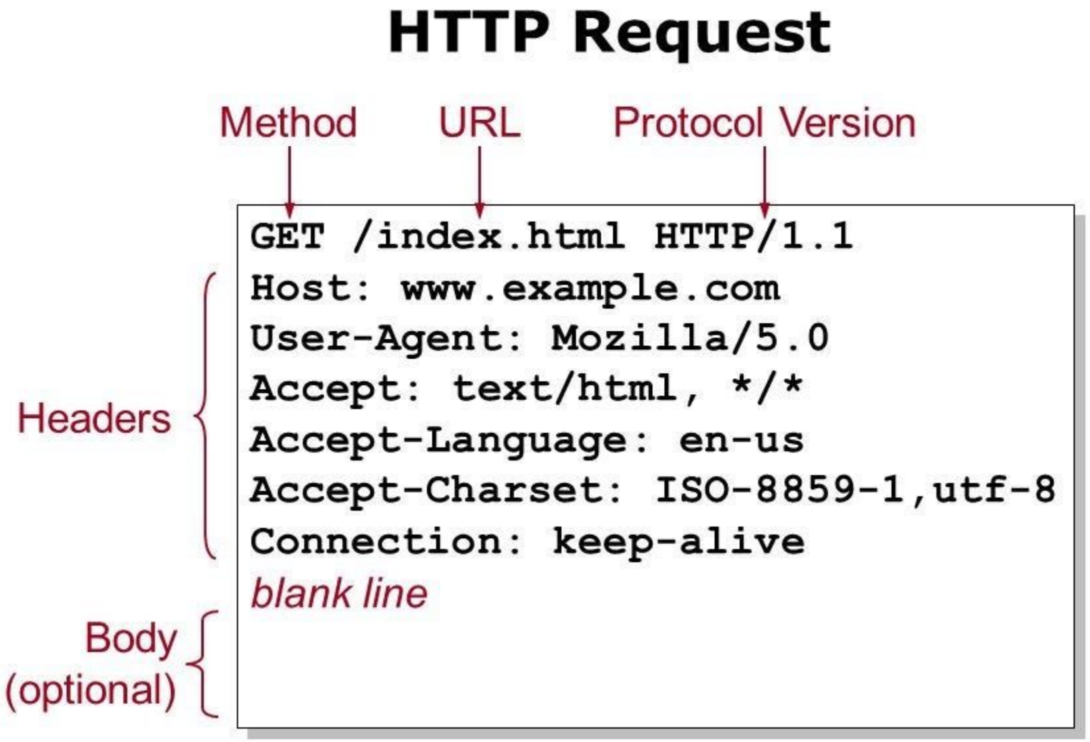
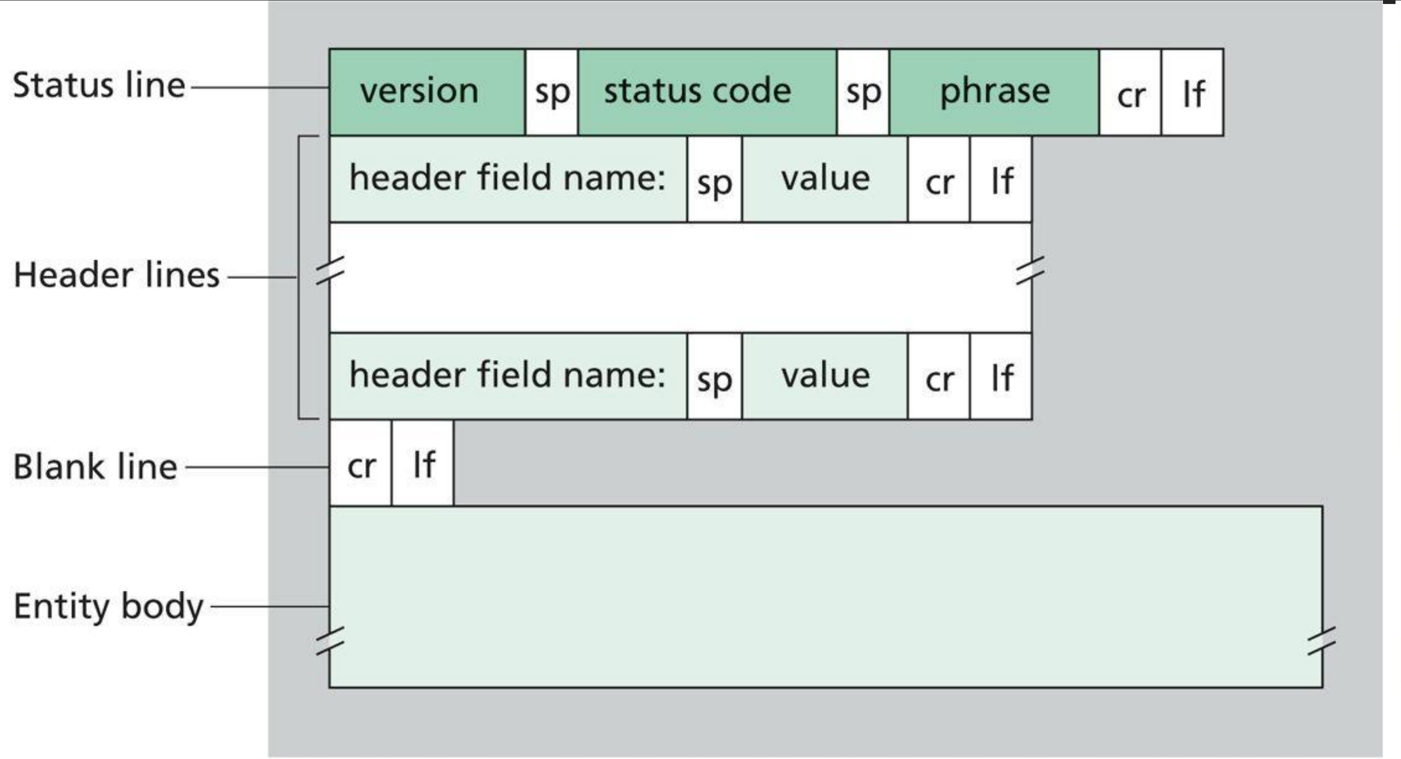


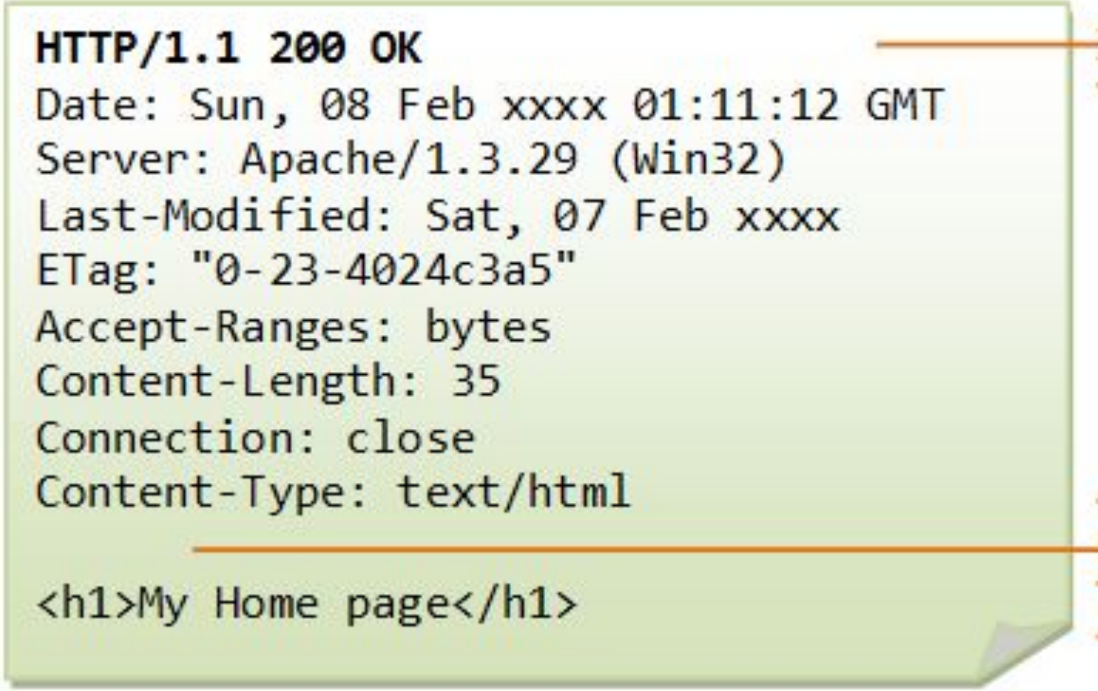
Figure 2.8 ♦ General format of a request message



```
GET /index.html HTTP/1.1
Host: www.example.com
User-Agent: Mozilla/5.0
Accept: text/html, */*
Accept-Language: en-us
Accept-Charset: ISO-8859-1,utf-8
Connection: keep-alive
```

HTTP 1.1 Response

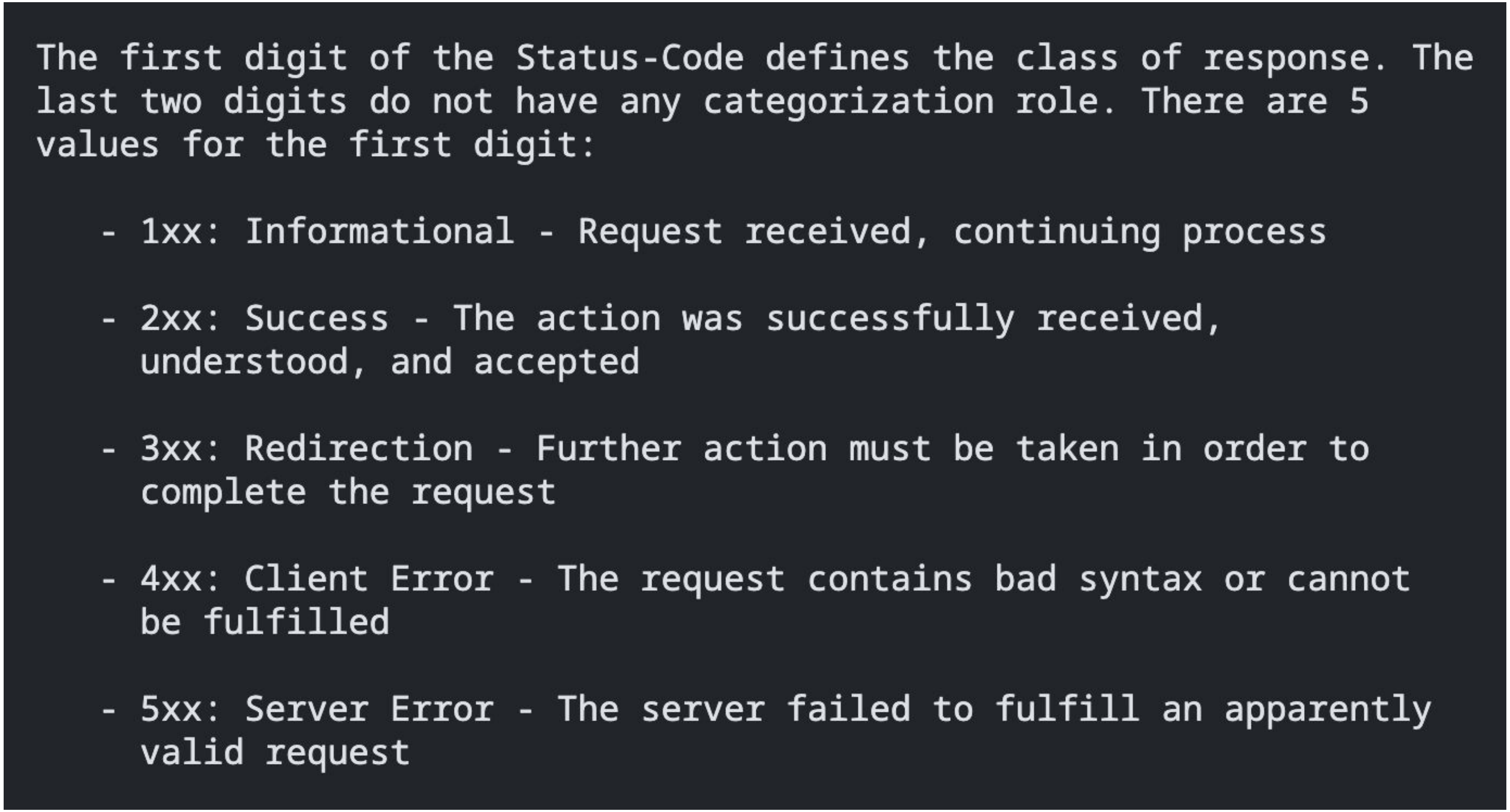




```
HTTP/1.1 200 OK
Date: Sun, 08 Feb xxxx 01:11:12 GMT
Server: Apache/1.3.29 (Win32)
Last-Modified: Sat, 07 Feb xxxx
ETag: "0-23-4024c3a5"
Accept-Ranges: bytes
Content-Length: 35
Connection: close
Content-Type: text/html

<h1>My Home page</h1>
```

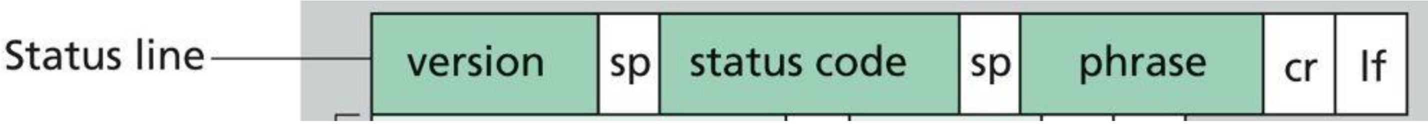
Figure 2.9 ♦ General format of a response message

Status Codes



```
The first digit of the Status-Code defines the class of response. The
last two digits do not have any categorization role. There are 5
values for the first digit:

   - 1xx: Informational - Request received, continuing process

   - 2xx: Success - The action was successfully received,
     understood, and accepted

   - 3xx: Redirection - Further action must be taken in order to
     complete the request

   - 4xx: Client Error - The request contains bad syntax or cannot
     be fulfilled

   - 5xx: Server Error - The server failed to fulfill an apparently
     valid request
```

A note on CR/LF



- **CR:** Carriage Return (**`0x0D`**) – often **`\r`**
- **LF:** Line Feed (**`0x0A`**) – often **`\n`**
- Windows traditionally uses CR LF, Unix uses just LF to break lines apart – will often see `\r\n` when printing out HTTP requests.
- Comes from typewriters – **LF** moves the paper up and **CR** returns the cursor to the leftmost character of the line.

Q1

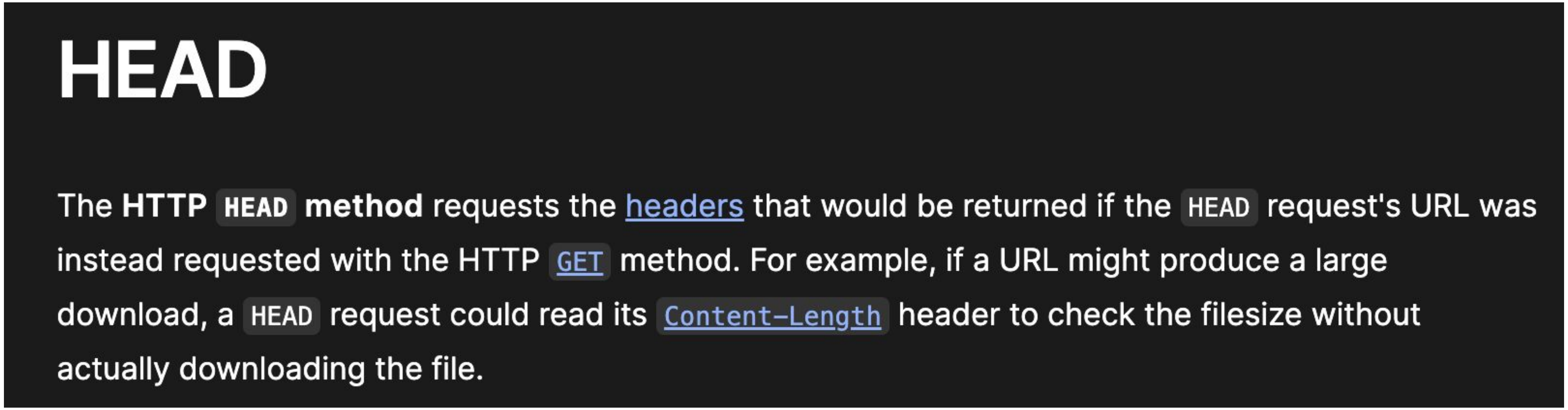
True or False: HTTP 1.1 response messages never have an empty message body.

Q1

HTTP response messages never have an empty message body.

False. Some HTTP response messages have an empty message body.

- E.g. A HEAD request does not require (and usually doesn’t have) a message body.
- E.g. A HEAD response **may not have** a message body.

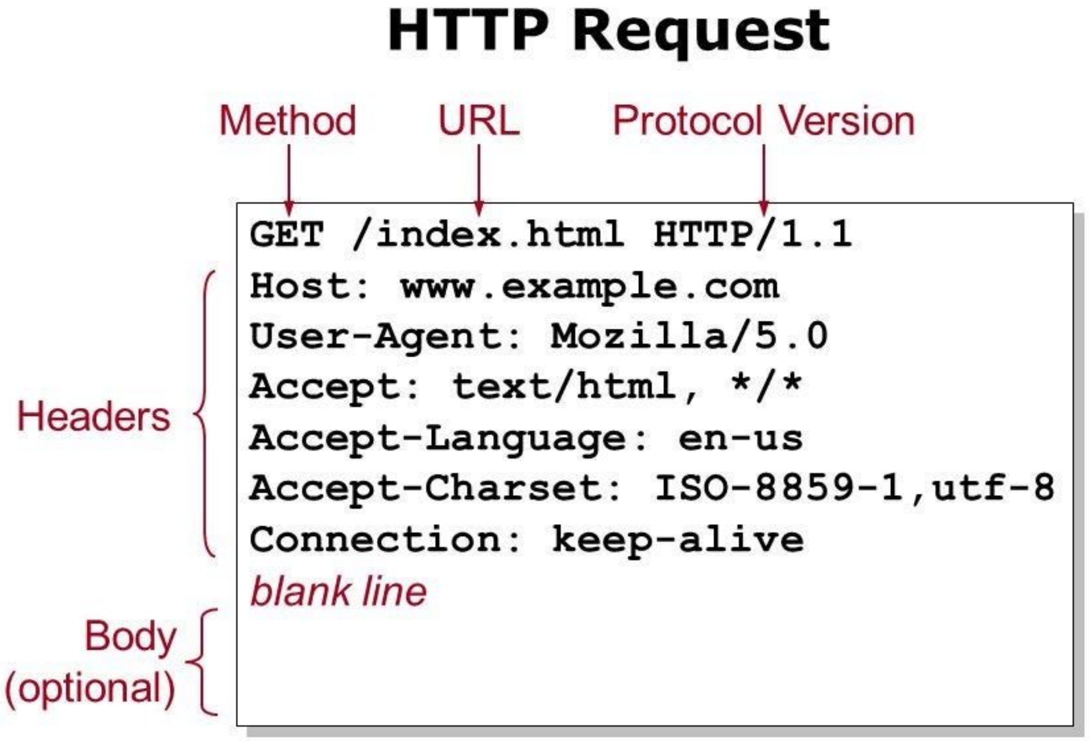


HEAD

The **HTTP `HEAD` method** requests the headers that would be returned if the `HEAD` request's URL was instead requested with the HTTP `GET` method. For example, if a URL might produce a large download, a `HEAD` request could read its `Content-Length` header to check the filesize without actually downloading the file.

Q2

True or False: HTTP 1.1 request messages never have an empty set of headers.

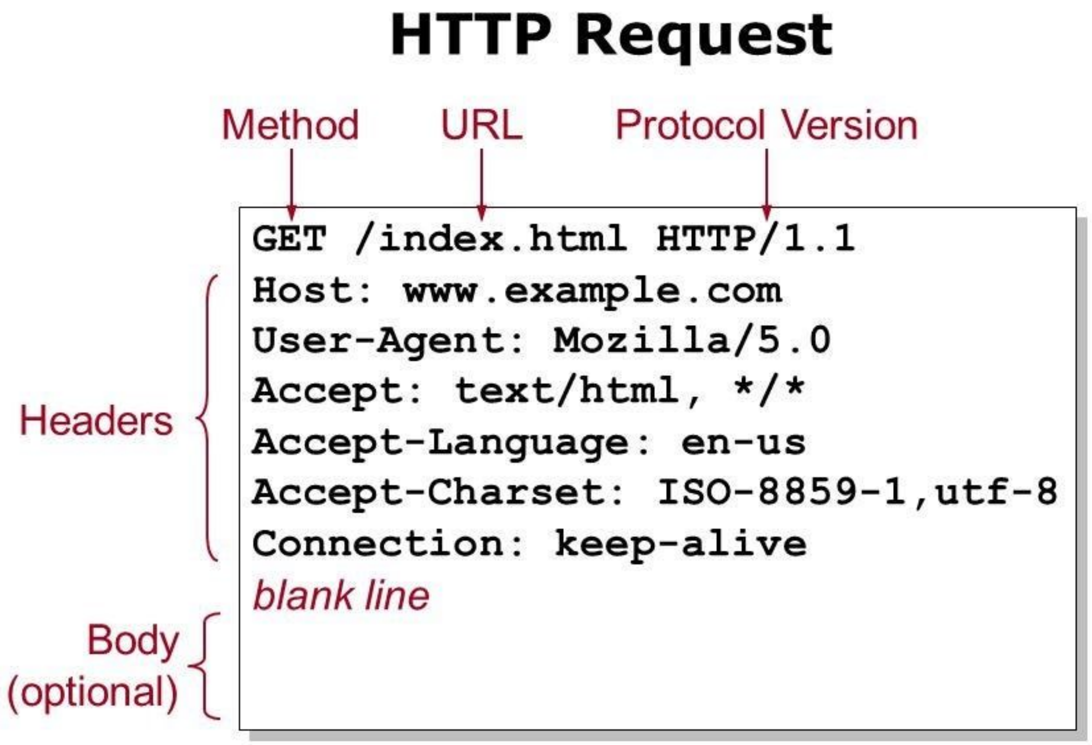


```
GET /index.html HTTP/1.1
Host: www.example.com
User-Agent: Mozilla/5.0
Accept: text/html, */*
Accept-Language: en-us
Accept-Charset: ISO-8859-1,utf-8
Connection: keep-alive
blank line
```

Q2

True or False: HTTP 1.1 request messages never have an empty set of headers.

True. For example, the **Host** header is required!



HTTP Request

Method URL Protocol Version

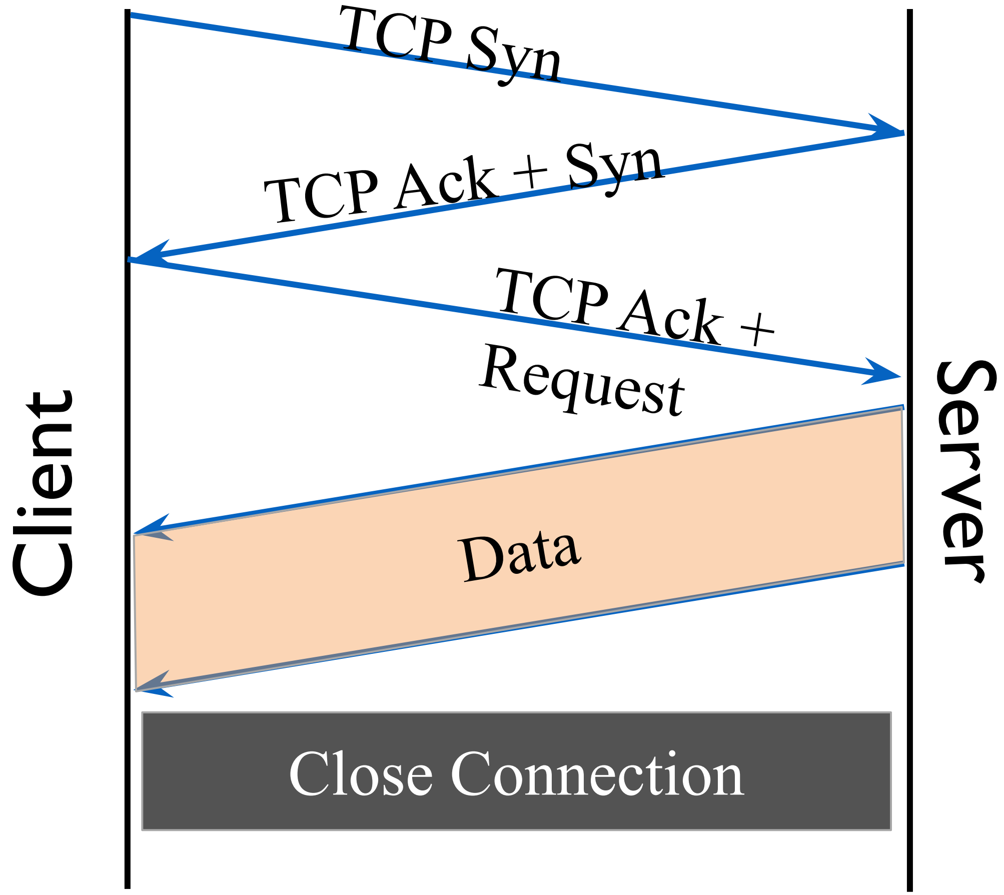
```
GET /index.html HTTP/1.1
Host: www.example.com
User-Agent: Mozilla/5.0
Accept: text/html, */*
Accept-Language: en-us
Accept-Charset: ISO-8859-1,utf-8
Connection: keep-alive
```

blank line

Headers

Body
(optional)

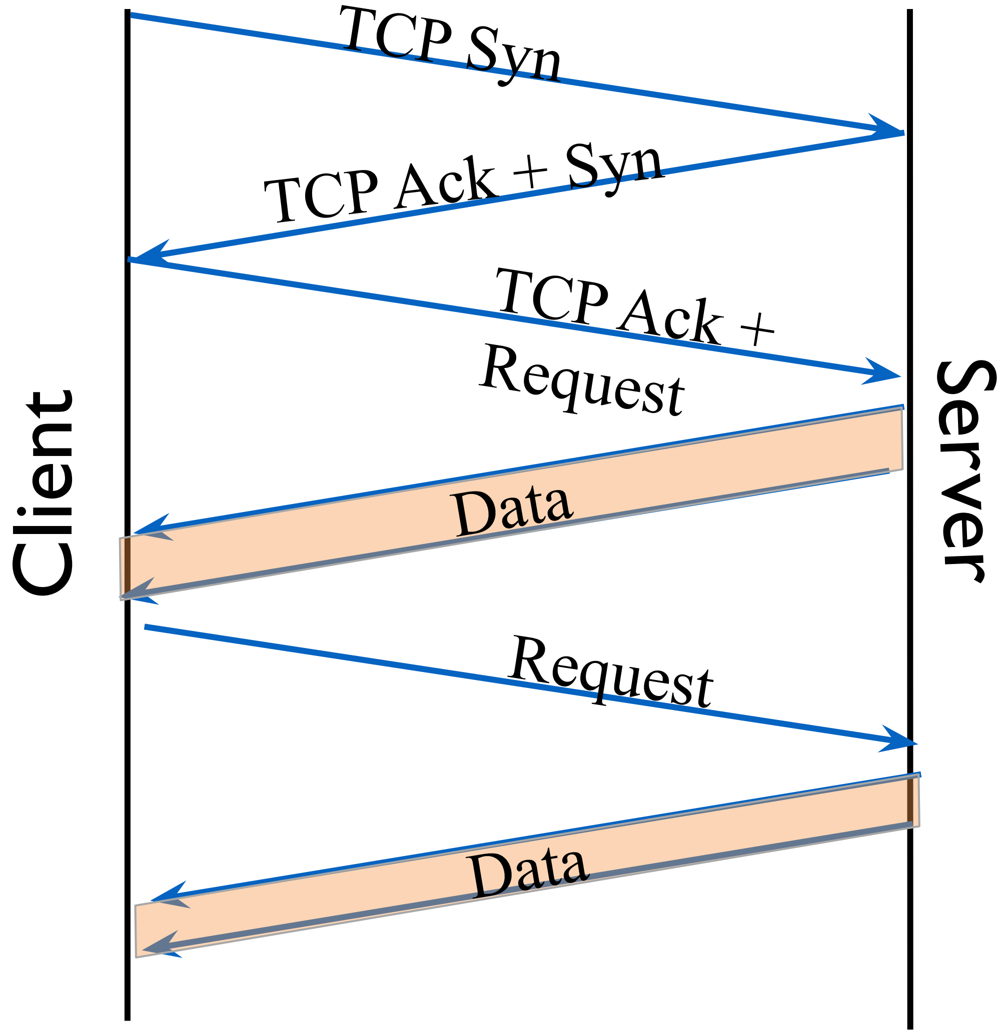
Persistency & Multiplexing



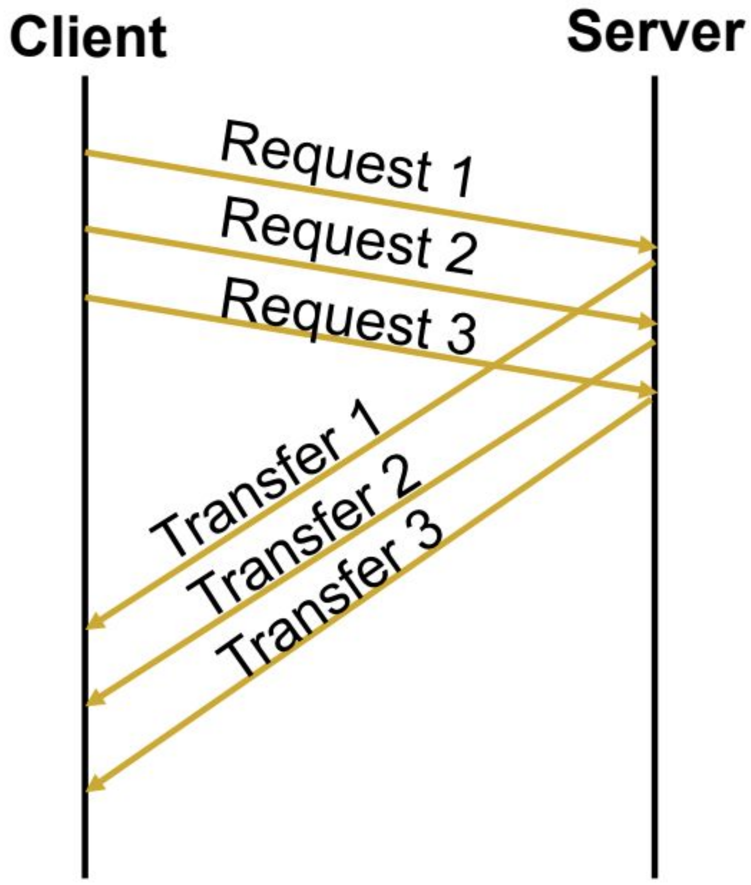
Non-Persistent Connection

- In **HTTP 1.0**, connections are non-persistent by default – they terminate after one piece of data is exchanged.
- In **HTTP 1.1**, connections are persistent by default, meaning that they aren’t closed after one request.
- In **HTTP 2**, connections can be multiplexed, which means that multiple requests can be sent concurrently.

Persistency & Multiplexed Connection



Persistent Connection



Multiplexed Connection

Q3

True or False: Two distinct Web pages (for example, umich.edu/research.html and umich.edu/students.html) can be sent over the same persistent connection.

Q3

True or False: Two distinct Web pages (for example, umich.edu/research.html and umich.edu/students.html) can be sent over the same persistent connection.

True. Both these pages are on the same physical server, so they can be retrieved on the same persistent connection.

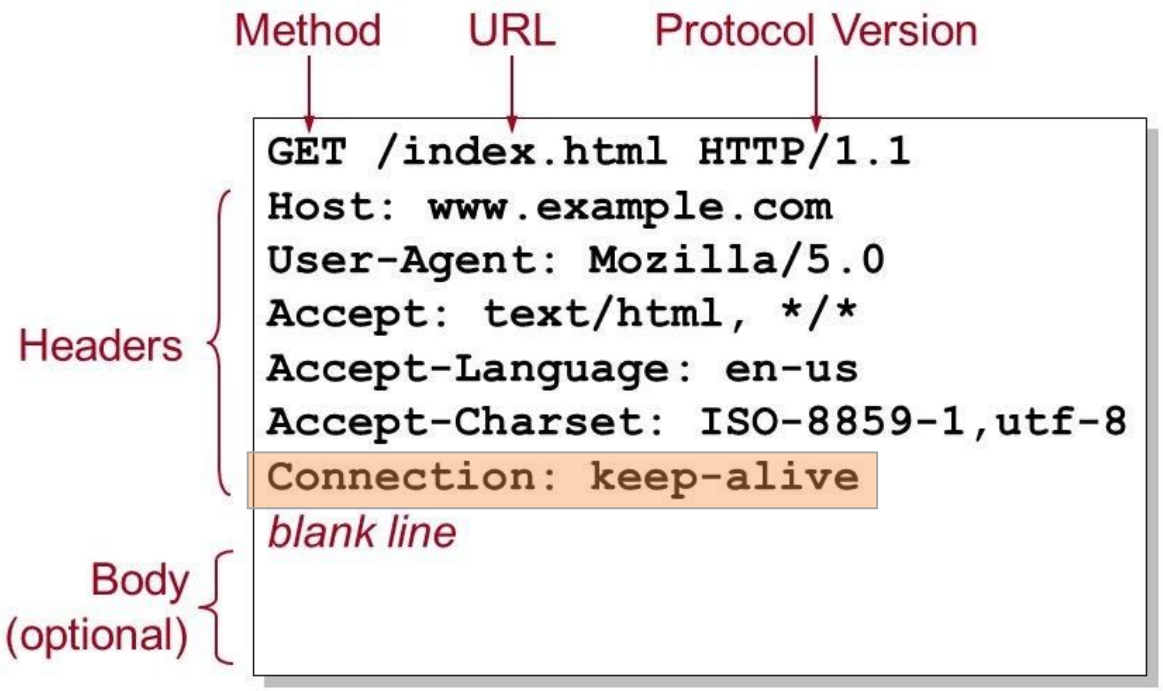
Q4

True or False: Any requests sent over HTTP 2 must be persistent.

Q4

True or False: Any requests sent over HTTP 2 must be persistent.

False. The persistency of the request is controlled by the `Connection` header, which is `keep-alive` by default (for a persistent connection), but can be set to `close` for a non-persistent connection.



```
GET /index.html HTTP/1.1
Host: www.example.com
User-Agent: Mozilla/5.0
Accept: text/html, */*
Accept-Language: en-us
Accept-Charset: ISO-8859-1,utf-8
Connection: keep-alive
```

Q5

True or False: Pipelining and multiplexing refer to the same thing.

Q5

True or False: Pipelining and multiplexing refer to the same thing.

False.

Pipelining

- Introduced in HTML 1.1, but not supported in most implementations of HTML 1.1.
- Allows for clients to make multiple requests at once.
- Servers must respond to requests in-order.
- Does not solve head-of-line blocking problem.

Multiplexing

- Introduced in HTML 2.
- Allows for clients to make multiple requests at once.
- Servers can respond to requests in any order they want.
- Solves head-of-line blocking problem!

Q6

You request a very small HTML file from a server. This HTML references **eight** other very small images. Let **X** denote the RTT between the localhost and the server. How much time elapses with **non-persistent HTTP with no parallel TCP connections**?

Q6

You request a very small HTML file from a server. This HTML references **eight** other very small images. Let **X** denote the RTT between the localhost and the server. How much time elapses with **non-persistent HTTP with no parallel TCP connections**?

2X + 8 * (2X) = **18X**

Q7

You request a very small HTML file from a server. This HTML references **eight** other very small images. Let **X** denote the RTT between the localhost and the server. How much time elapses with **non-persistent HTTP with the browser configured for 5 parallel connections**?

Q7

You request a very small HTML file from a server. This HTML references **eight** other very small images. Let **X** denote the RTT between the localhost and the server. How much time elapses with **non-persistent HTTP with the browser configured for 5 parallel connections**?

```
2X + CEIL(8 / 5) * (2X) = 2X + 2 * 2X = 6X
```

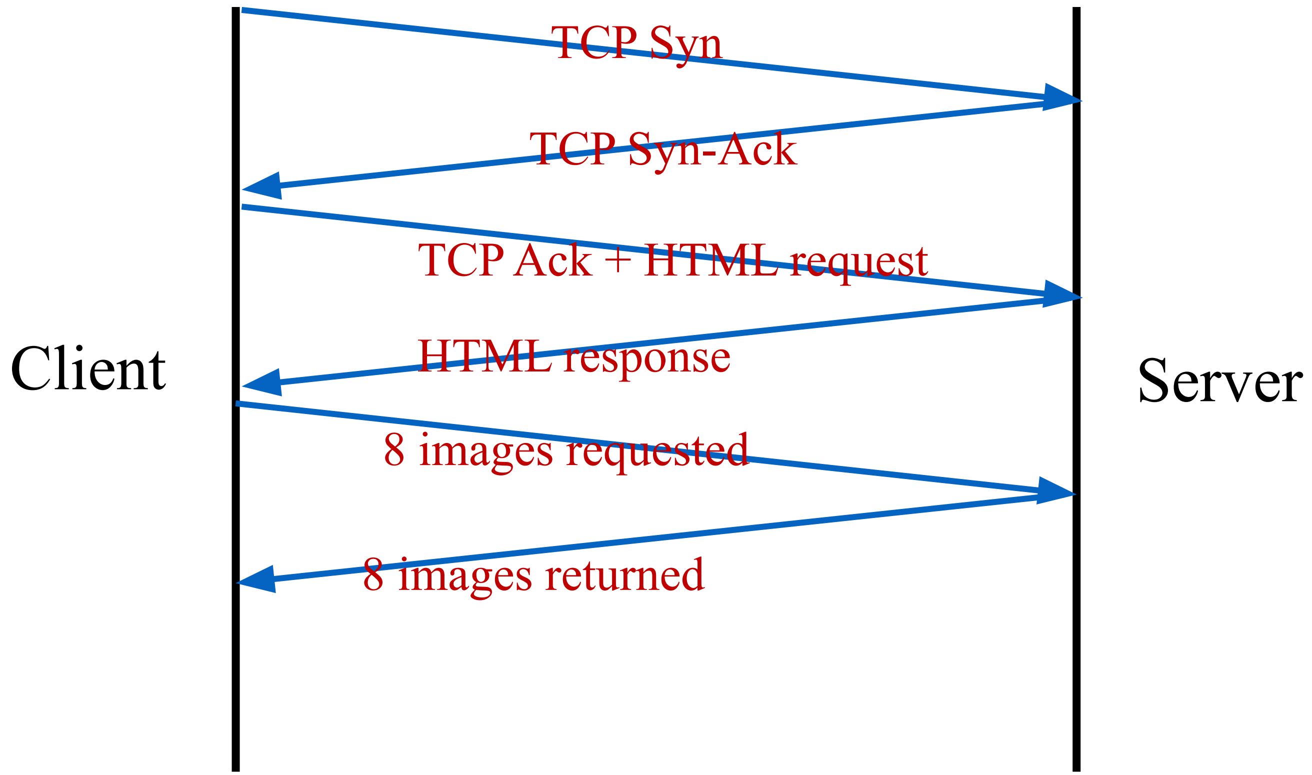
Q8

You request a very small HTML file from a server.This HTML references eight other very small images. Let X denote the RTT between the localhost and the server. How much time elapses with **persistent HTTP with multiplexing**?

Q8

You request a very small HTML file from a server.This HTML references eight other very small images. Let X denote the RTT between the localhost and the server. How much time elapses with **persistent HTTP with multiplexing**?

X + 2X = **3X**


TCP Syn
TCP Syn-Ack
TCP Ack + HTML request
HTML response
8 images requested
8 images returned
Client
Server