

CSCI 466: Networks

Lecture 5: Application Layer and HTTP

Reese Pearsall
Fall 2023

We will be working in **Wireshark** on Friday

What is the purpose of the **Network Core**?

- To route and transfer data to the correct destination ☒
- To ensure hosts can connect with one another ☐
- To connect users to a higher-tier ISP ☐
- ➔ All of the above ☒

The system of rules that define the format for how devices should communicate on the internet is known as what?

- ➔ A protocol ☒
- The OSI model ☐
- Physical Medium ☐
- Access Network ☐



Application Layer

Presentation Layer *

Session Layer *

Transport Layer

Network Layer

Data Link Layer

Physical Layer

OSI Model

Application Layer

Messages from Network Applications



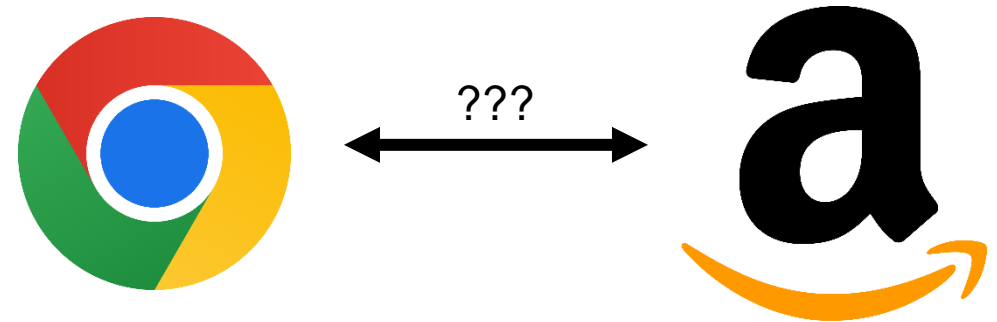
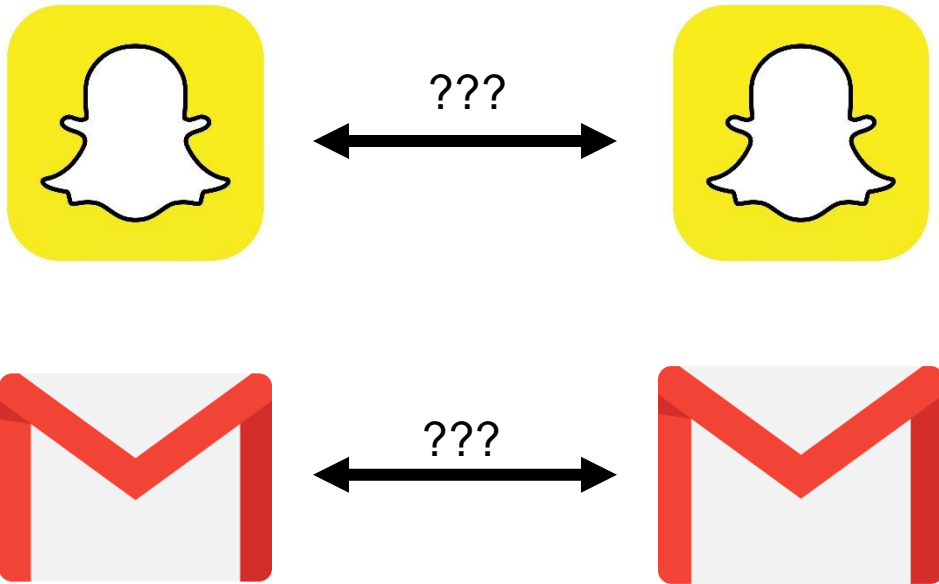
Physical Layer

Bits being transmitted over a copper wire

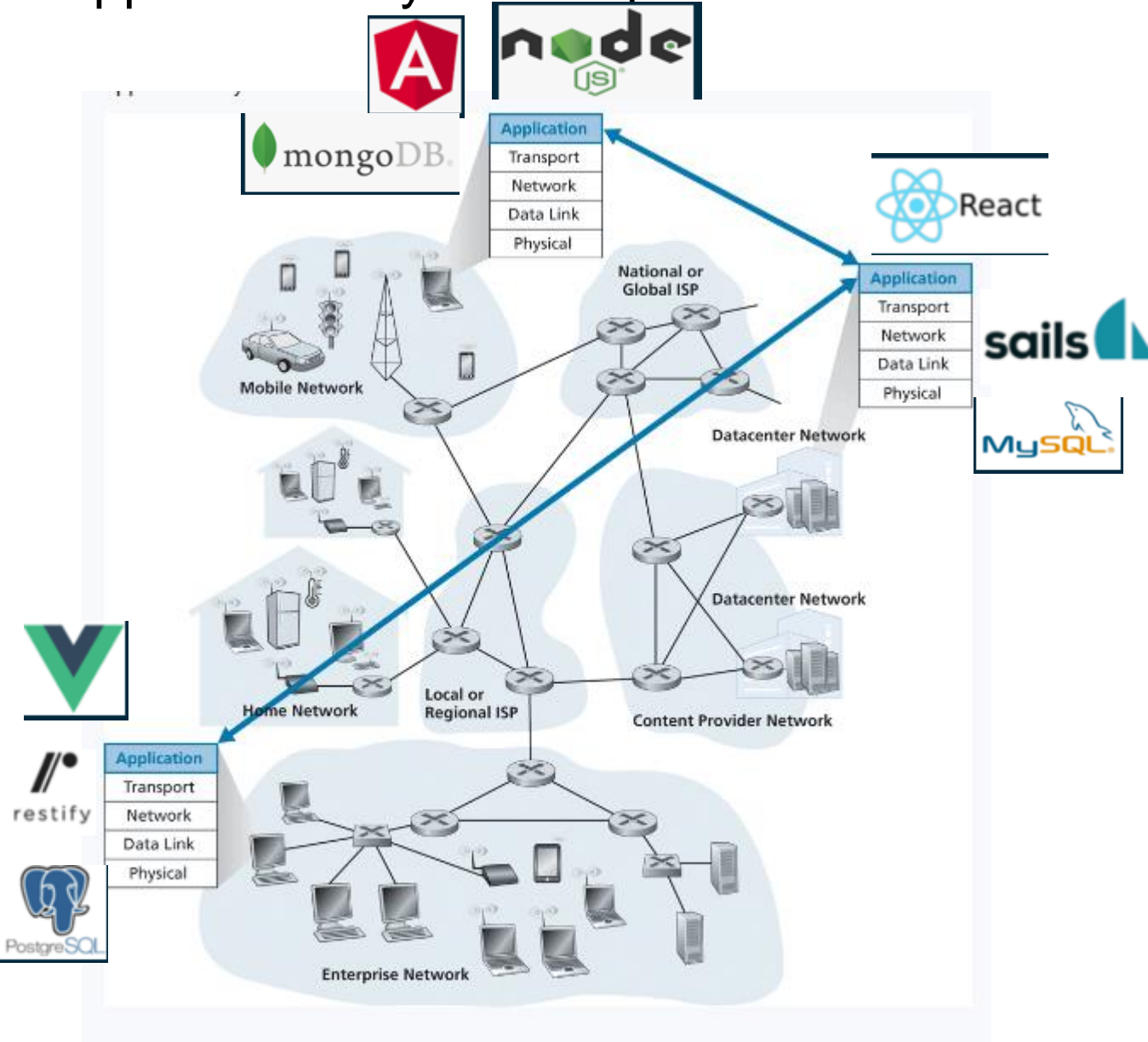
**In the textbook, they condense it to a 5-layer model, but 7 layers is what is most used*

Application Layer

OSI Model



Application Layer Principles

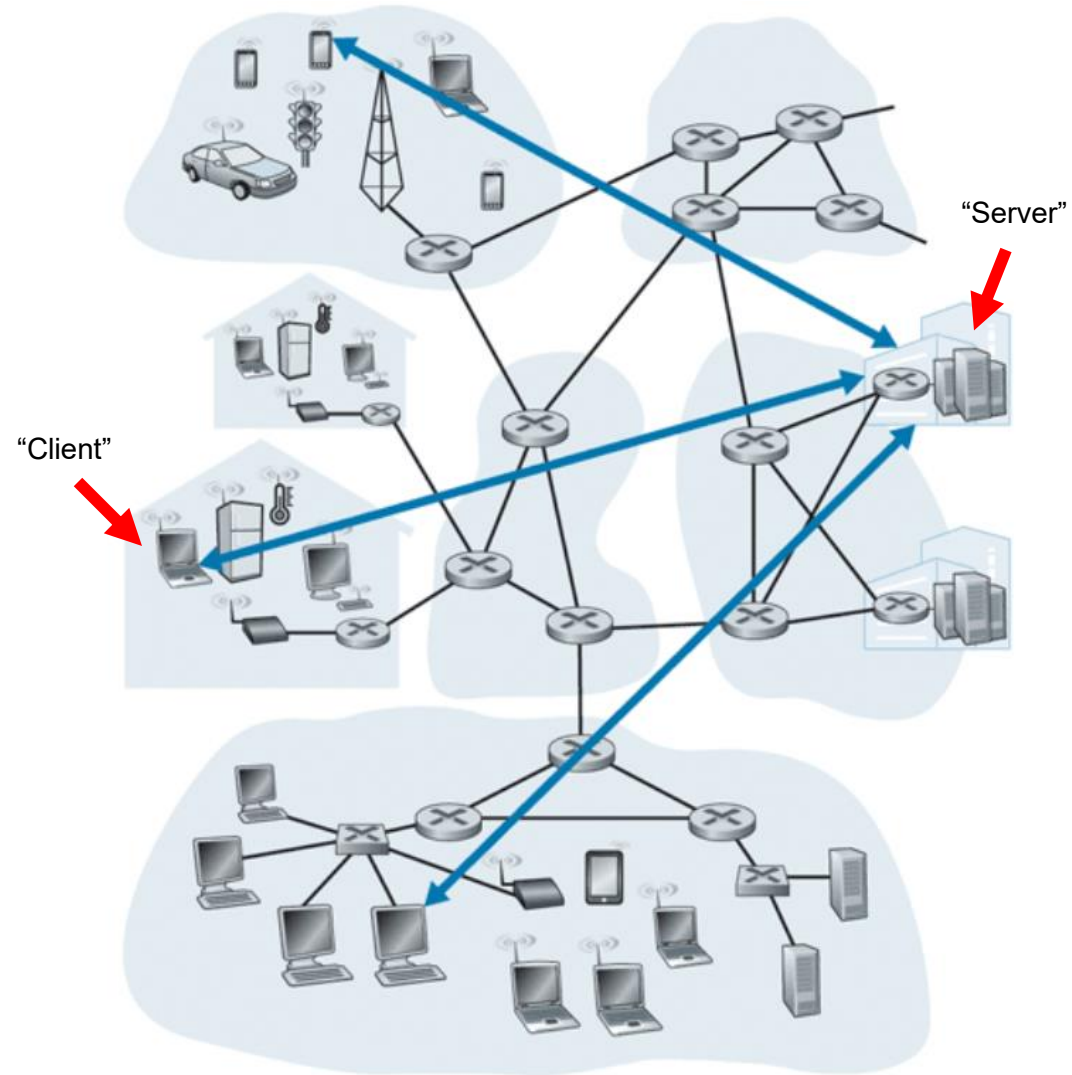


Web applications are built with different technologies

We need a universal method for communicating between applications connected to the internet

New web applications need to be “compatible” with other web applications they communicate with

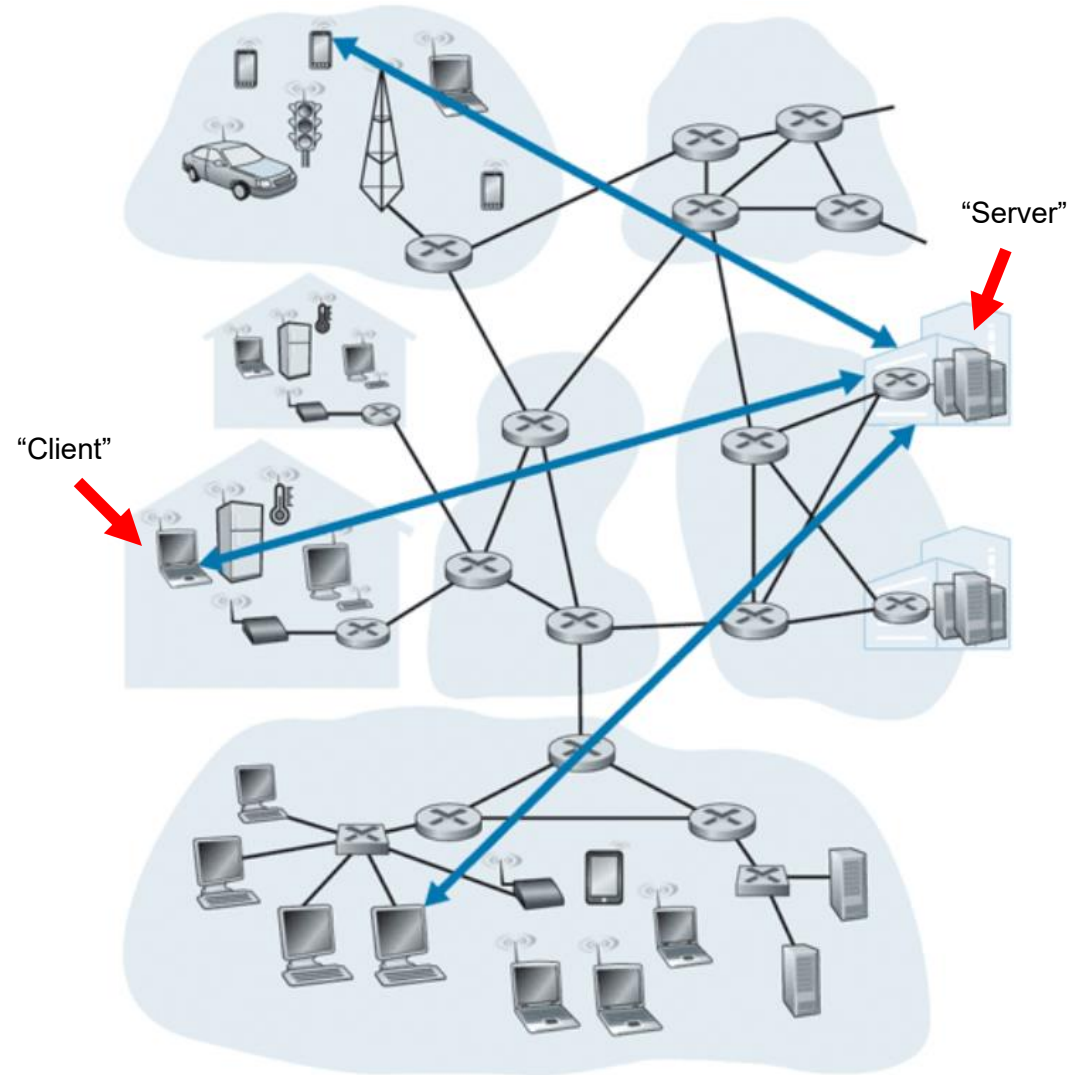
Web Application Architectures



Client-server architecture

Clients do not directly interact with each other

Web Application Architectures

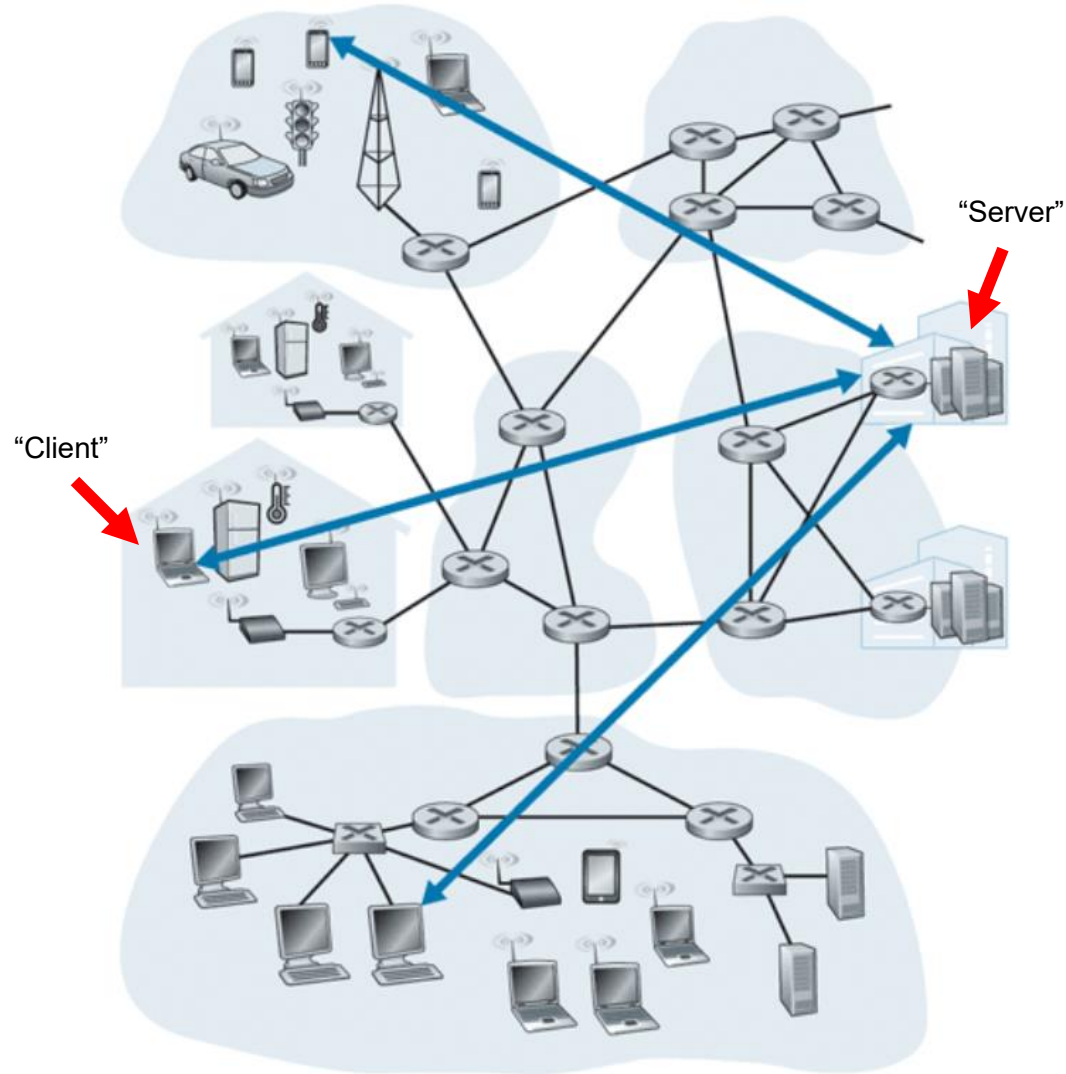


Client-server architecture

Clients do not directly interact with each other

ie. My web browser does not directly interact with your web browser

Web Application Architectures



Client-server architecture

Clients do not directly interact with each other

ie. My web browser does not directly interact with your web browser

Communication is done through a **Server**

- Online 24/7*
- Hosted in a **data center**



Web Application Architectures






Application Requirements

What type of services do different applications need from the network?

Application	Data Loss	Throughput	Time Sensitive

Application Requirements

What type of services do different applications need from the network?

Application	Data Loss	Throughput	Time Sensitive
			
	<div>Can the application tolerate potential loss of data?</div>	<div>Does the application require a specific bandwidth/throughput value?</div>	<div>Do endpoints need to receive data within X amount of time? Is delay allowed?</div>

Application Requirements

What type of services do different applications need from the network?

Application	Data Loss	Throughput	Time Sensitive
File Transfer	No loss	Elastic	No

Application Requirements

What type of services do different applications need from the network?

Application	Data Loss	Throughput	Time Sensitive
File Transfer	No loss	Elastic	No
Email	No loss	Elastic	No

Application Requirements

What type of services do different applications need from the network?

Application	Data Loss	Throughput	Time Sensitive
File Transfer	No loss	Elastic	No
Email	No loss	Elastic	No
Web Browsing	No loss	Elastic	No → Yes

Application Requirements

What type of services do different applications need from the network?

Application	Data Loss	Throughput	Time Sensitive
File Transfer	No loss	Elastic	No
Email	No loss	Elastic	No
Web Browsing	No loss	Elastic	No → Yes
Real-Time Audio	Loss-Tolerant	Inelastic	Yes

Application Requirements

What type of services do different applications need from the network?

Application	Data Loss	Throughput	Time Sensitive
File Transfer	No loss	Elastic	No
Email	No loss	Elastic	No
Web Browsing	No loss	Elastic	No → Yes
Real-Time Audio	Loss-Tolerant	Inelastic	Yes
Stored Video	Loss-Tolerant	Elastic	No

Application Requirements

What type of services do different applications need from the network?

Application	Data Loss	Throughput	Time Sensitive
File Transfer	No loss	Elastic	No
Email	No loss	Elastic	No
Web Browsing	No loss	Elastic	No → Yes
Real-Time Audio	Loss-Tolerant	Inelastic	Yes
Stored Video	Loss-Tolerant	Elastic	No
Interactive Games	Loss-Tolerant	Inelastic	Yes

Application Requirements

What type of services do different applications need from the network?

Application	Data Loss	Throughput	Time Sensitive
File Transfer	No loss	Elastic	No
Email	No loss	Elastic	No
Web Browsing	No loss	Elastic	No → Yes
Real-Time Audio	Loss-Tolerant	Inelastic	Yes
Stored Video	Loss-Tolerant	Elastic	No
Interactive Games	Loss-Tolerant	Inelastic	Yes
Text Messaging	No loss	Elastic	No

Application Requirements

What type of services do different applications need from the network?

Application	Data Loss	Throughput	Time Sensitive	Security
File Transfer	No loss	Elastic	No	???
Email	No loss	Elastic	No	Need Security
Web Browsing	No loss	Elastic	No → Yes	Need Security
Real-Time Audio	Loss-Tolerant	Inelastic	Yes	???
Stored Video	Loss-Tolerant	Elastic	No	???
Interactive Games	Loss-Tolerant	Inelastic	Yes	Don't need security
Text Messaging	No loss	Elastic	No	Need Security



User Datagram Prot. (UDP)

Unreliable data transfer

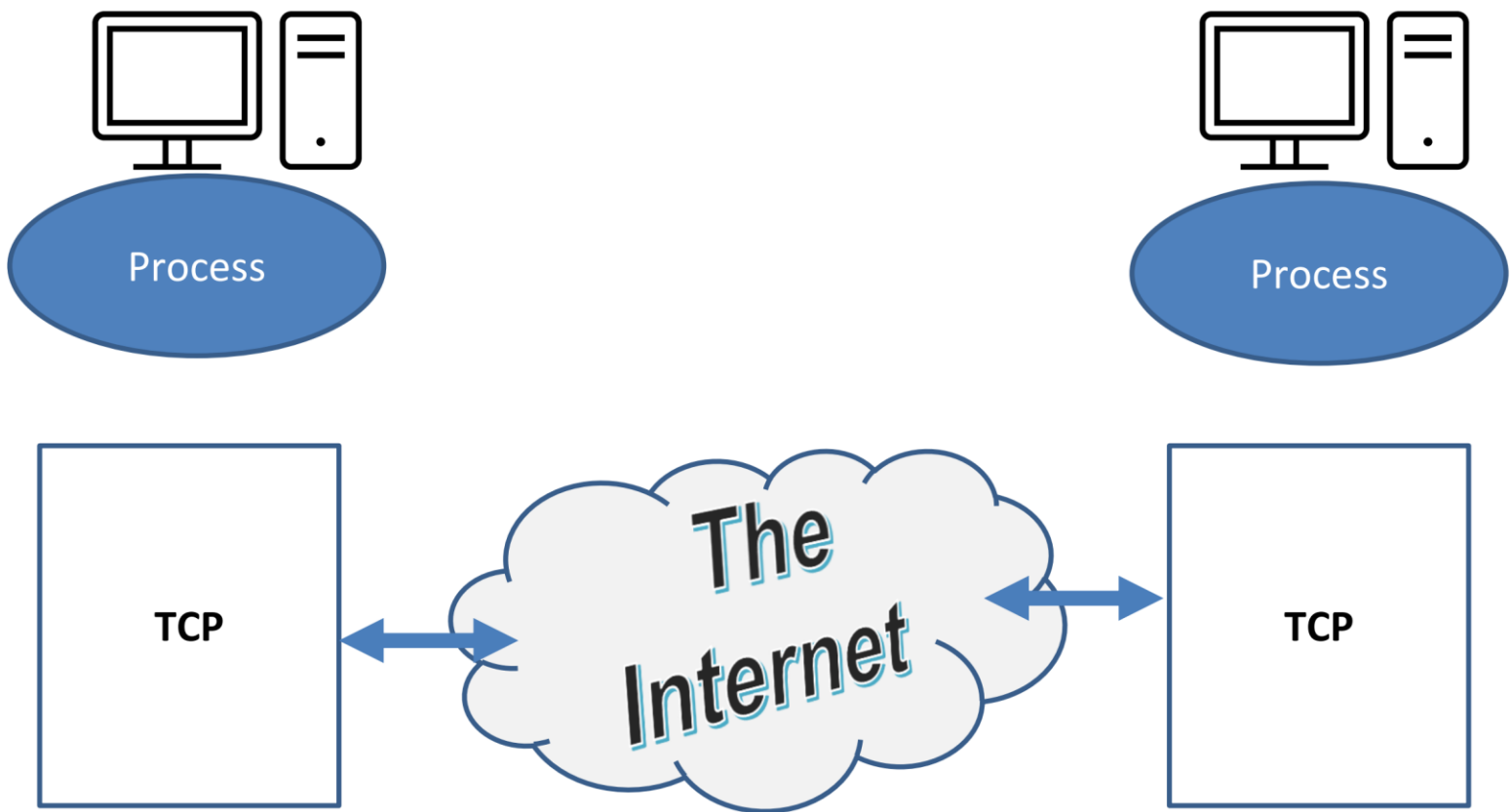
- Connection-less
 - Don't know if receiver is present
- No flow control
 - Overflow at receiver possible
- No congestion control
 - Sender can overload the network
- No guarantees on
 - End-to-end delay
 - Throughput
 - Security

Transmission Control Prot. (TCP)

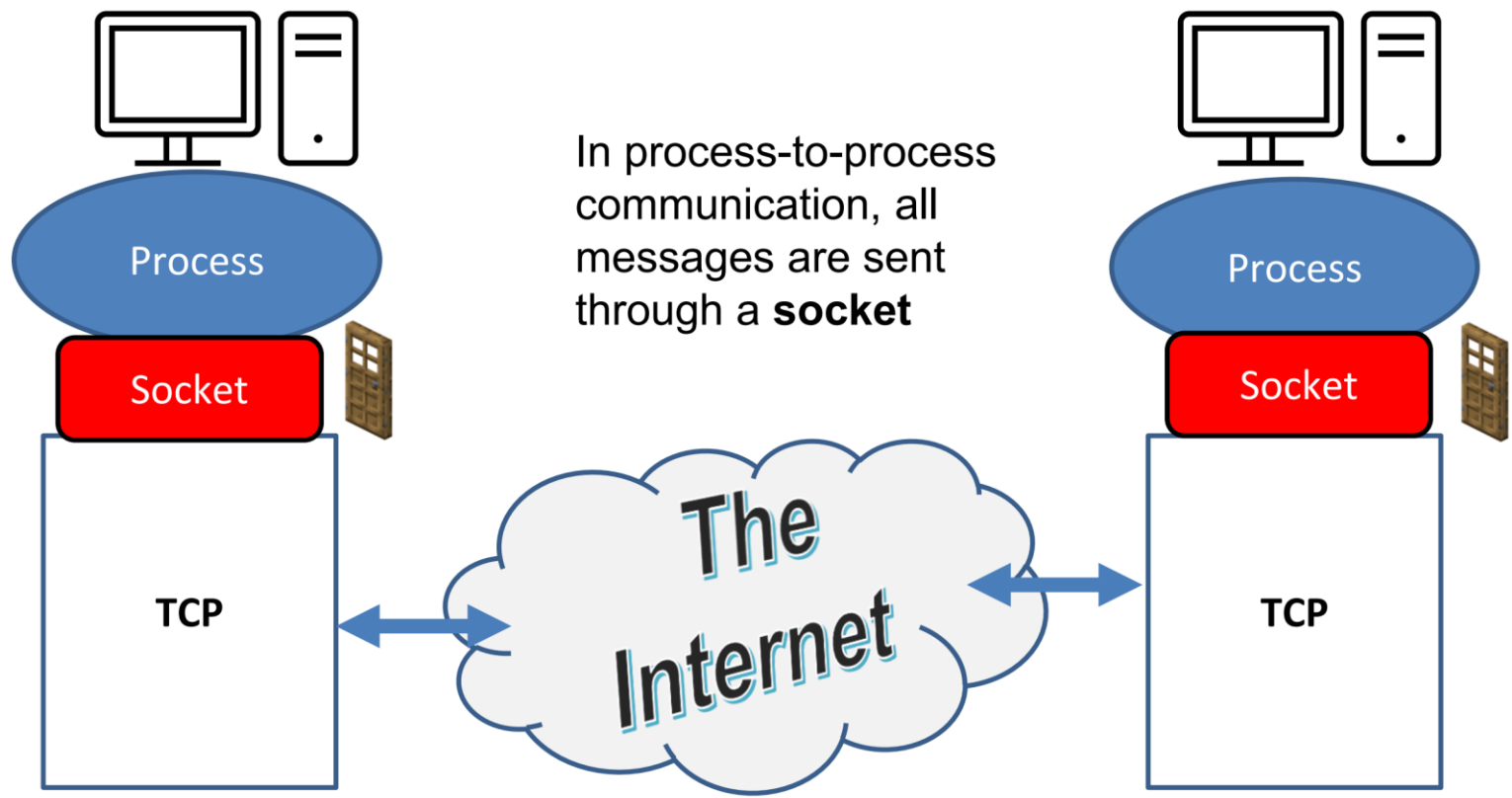
Reliable stream transport

- Connection-oriented
 - Establishes receiver presence
- Flow control
 - Sender won't overwhelm receiver
- Congestion control
 - Senders won't overload network
- No guarantees on
 - End-to-end delay
 - Throughput
 - Security

Application to Transport Interface

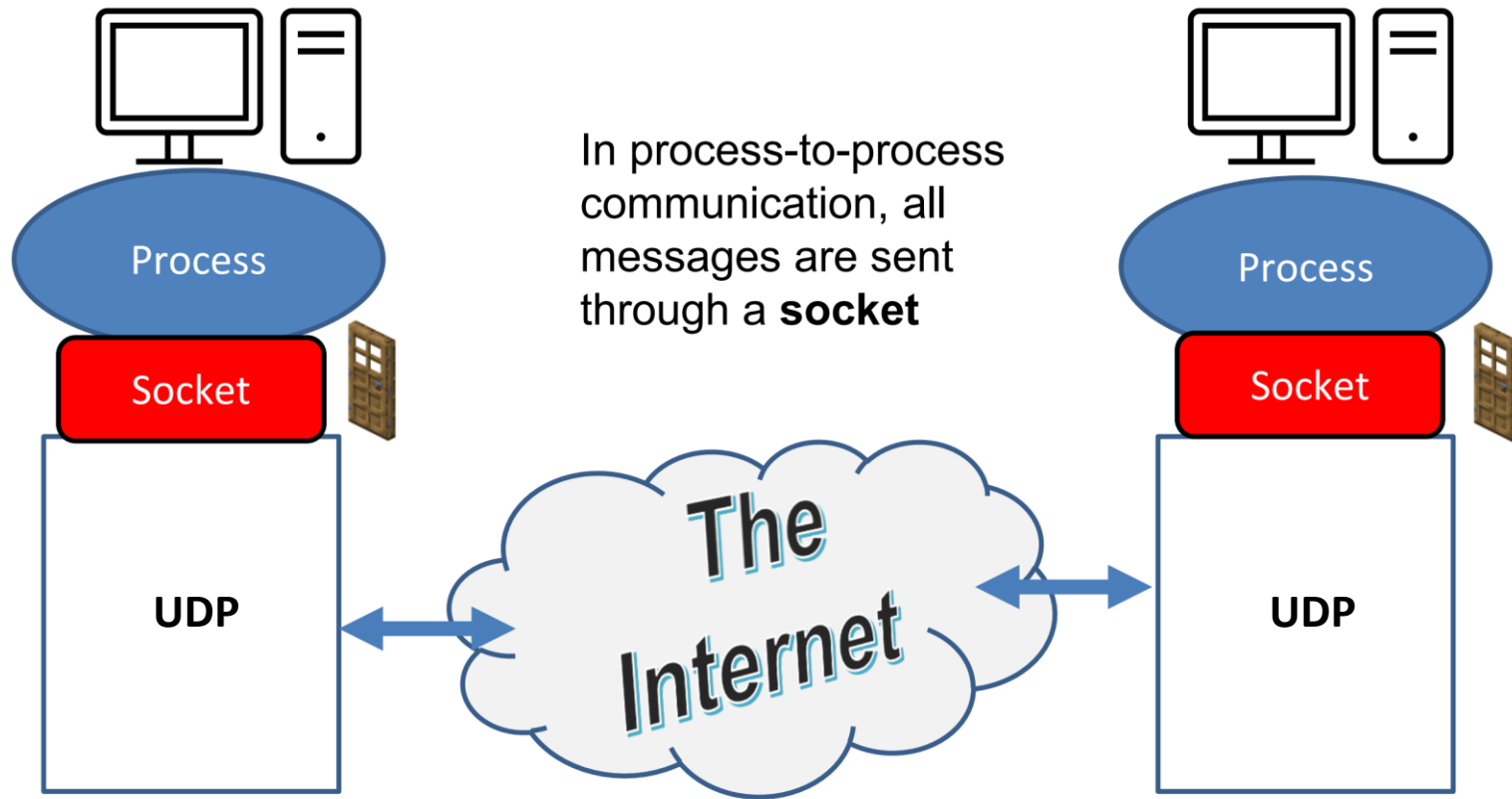


Application to Transport Interface



Option 1: Application sends messages through a **TCP** socket

Application to Transport Interface



Option 2: Application sends messages through a **UDP** socket

HyperText Transfer Protocol (HTTP)- protocol that dictates the transmitting of hypermedia documents, such as **HTML** and other webpage objects

“The language of the web”

HyperText Transfer Protocol (HTTP)- protocol that dictates the transmitting of hypermedia documents, such as **HTML** and other webpage objects

“The language of the web”

Uniform Resource Locator (URL)- Addressing scheme for web objects

`scheme://domain:port/path_to_object?query_string`

HyperText Transfer Protocol (HTTP)- protocol that dictates the transmitting of hypermedia documents, such as **HTML** and other webpage objects

“The language of the web”

Uniform Resource Locator (URL)- Addressing scheme for web objects

`scheme://domain:port/path_to_object?query_string`

`http://www.cs.montana.edu/pearsall/classes/fall2023/466/main.html`

HyperText Transfer Protocol (HTTP)- protocol that dictates the transmitting of hypermedia documents, such as **HTML** and other webpage objects

“The language of the web”

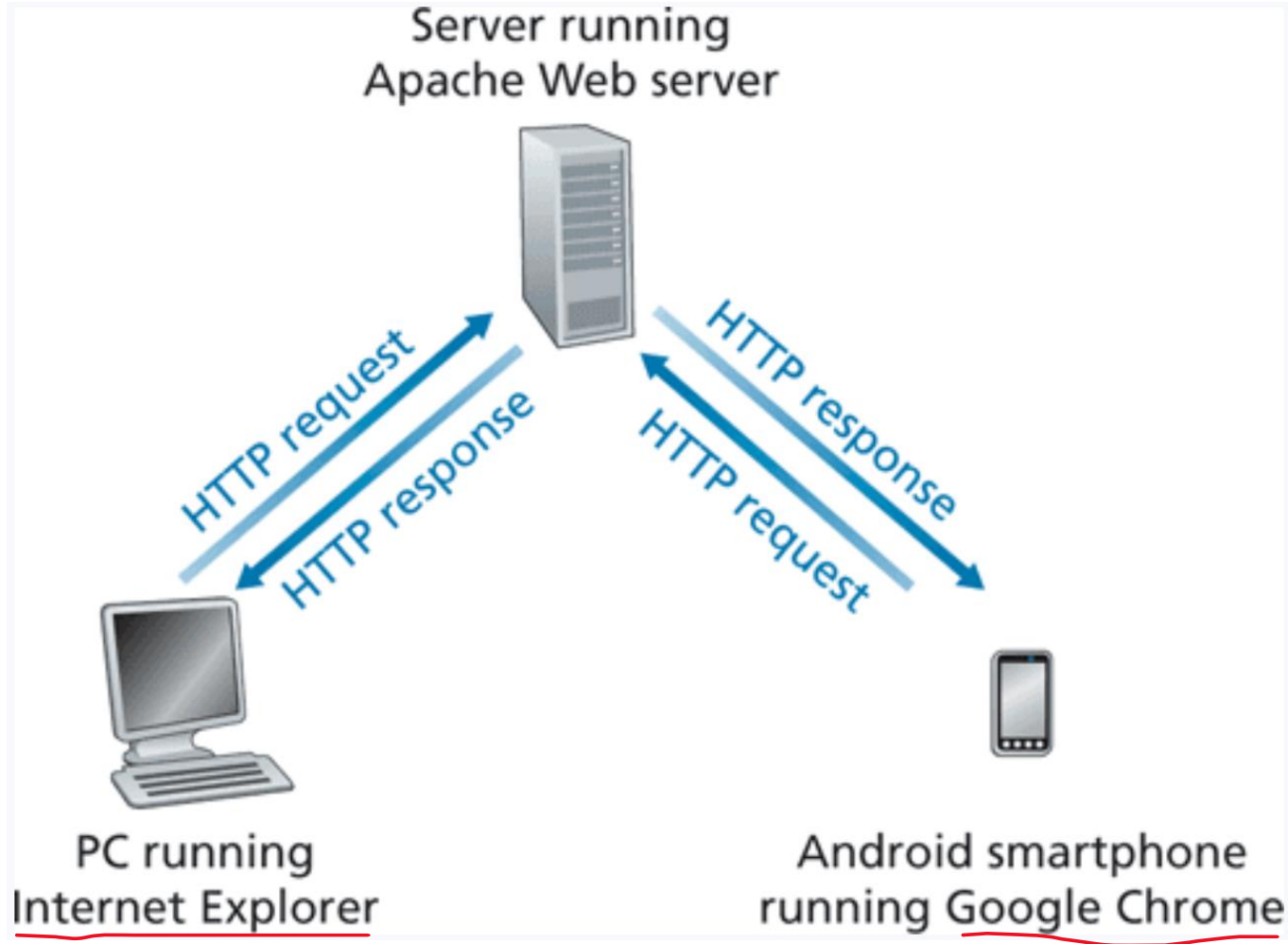
Uniform Resource Locator (URL)- Addressing scheme for web objects

scheme://domain:port/path_to_object?query_string

http://www.cs.montana.edu/pearsall/classes/fall2023/466/main.html

*Web object
that is
retrieved!*

HTTP



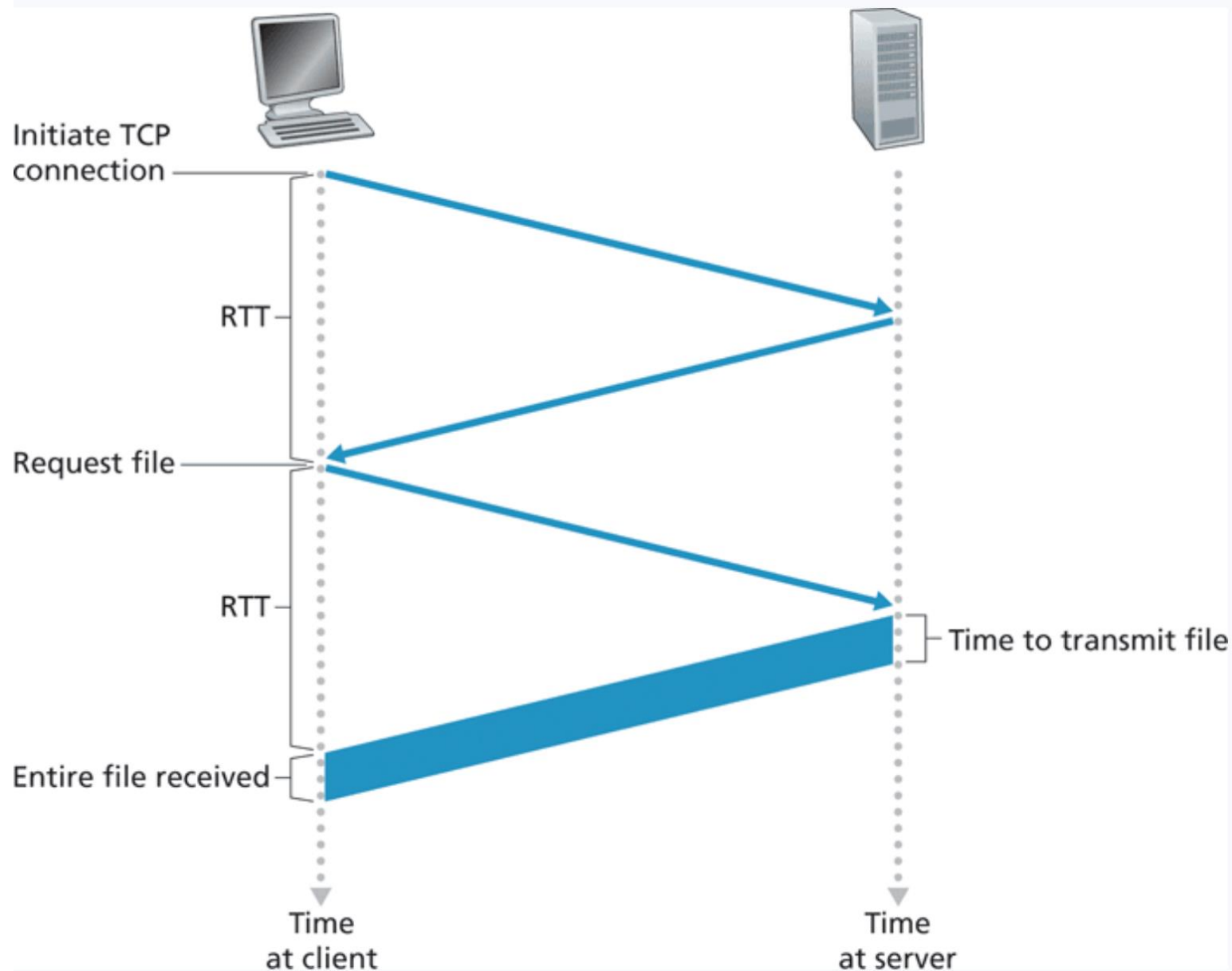
HTTP is *stateless* protocol that acts as a uniform API for different platforms

Built on **TCP**

HTTP protocol consists of two important pieces

- **HTTP Request**
- **HTTP Response**

HTTP (Non Persistent)

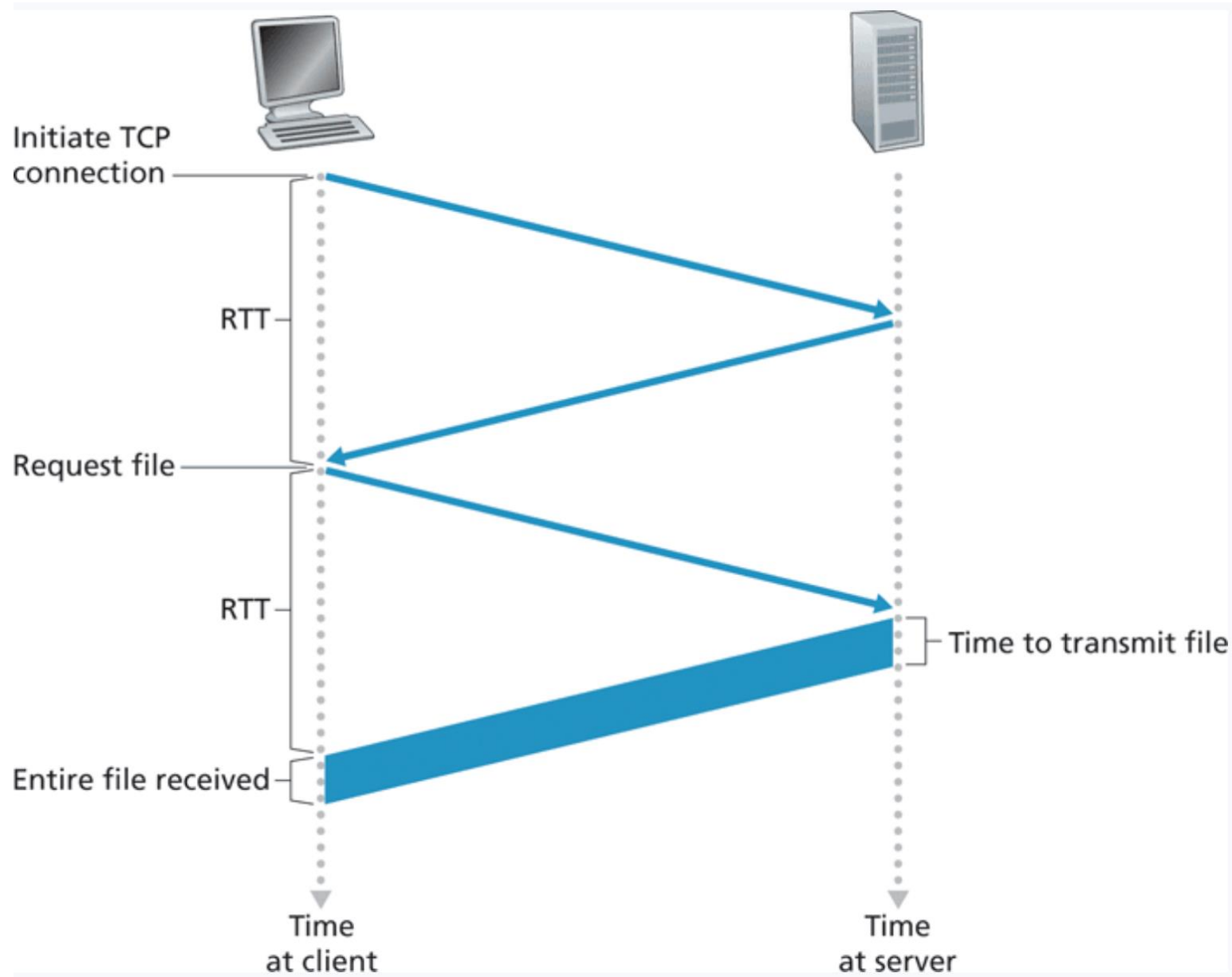


HTTP can either be **persistent** or **non persistent**

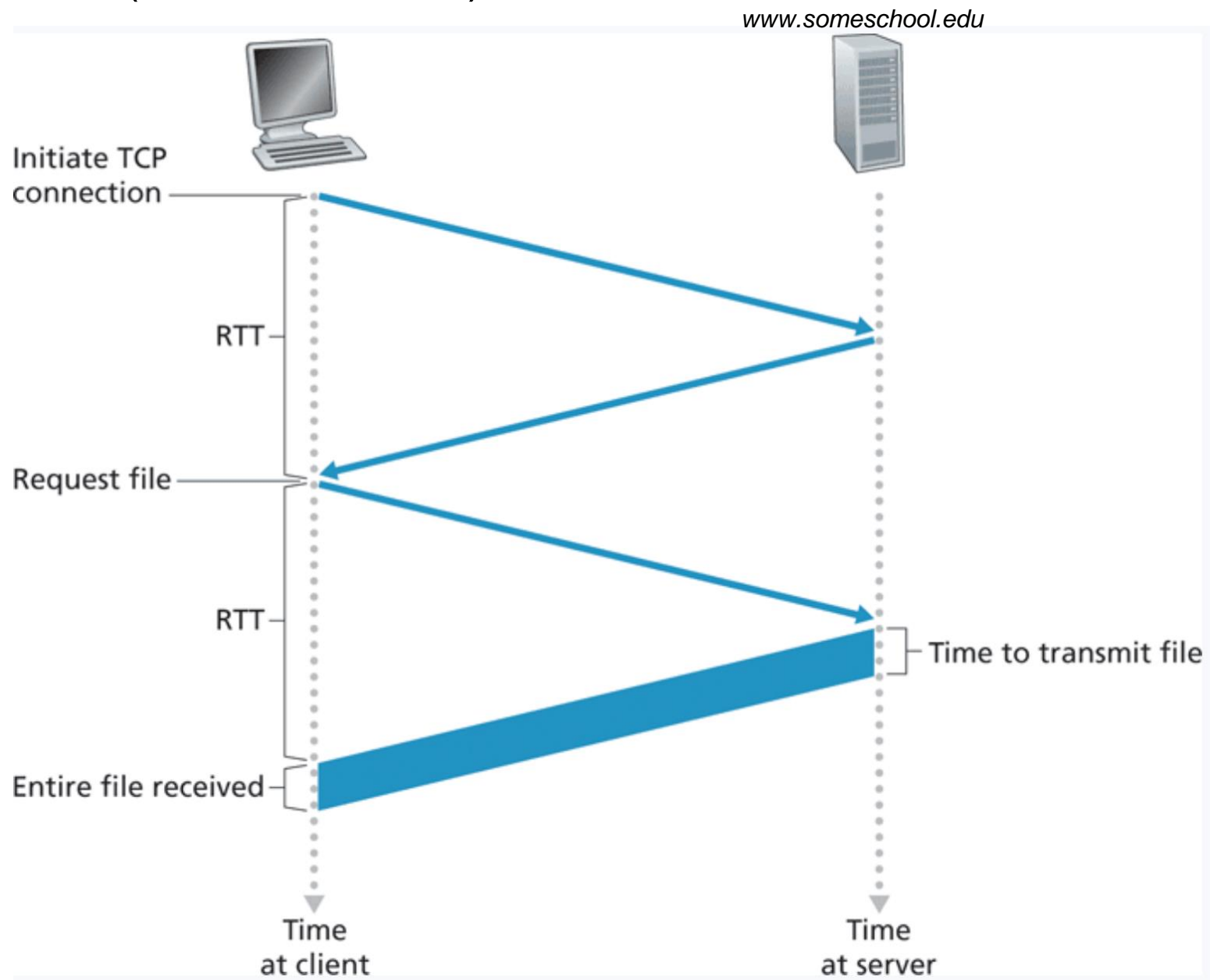
Persistent- the same TCP connection is used to transmit all web objects

Non persistent- a new TCP connection is opened for each web object

HTTP (Non Persistent) *HTTP 1.0*

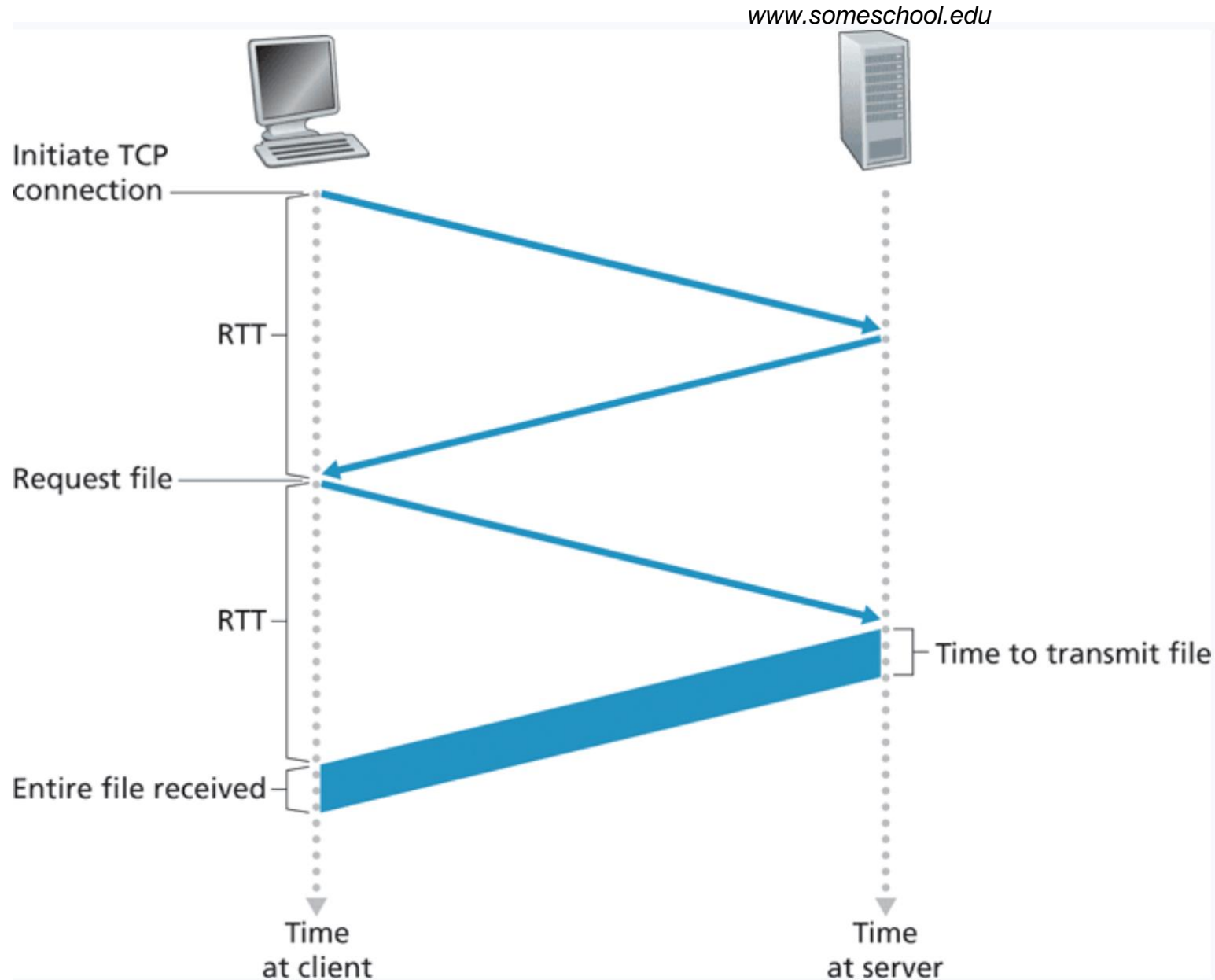


HTTP (Non Persistent) HTTP 1.0



1. HTTP client process initiates a TCP connection to the server on port number 80. A socket is created at both the client and the server

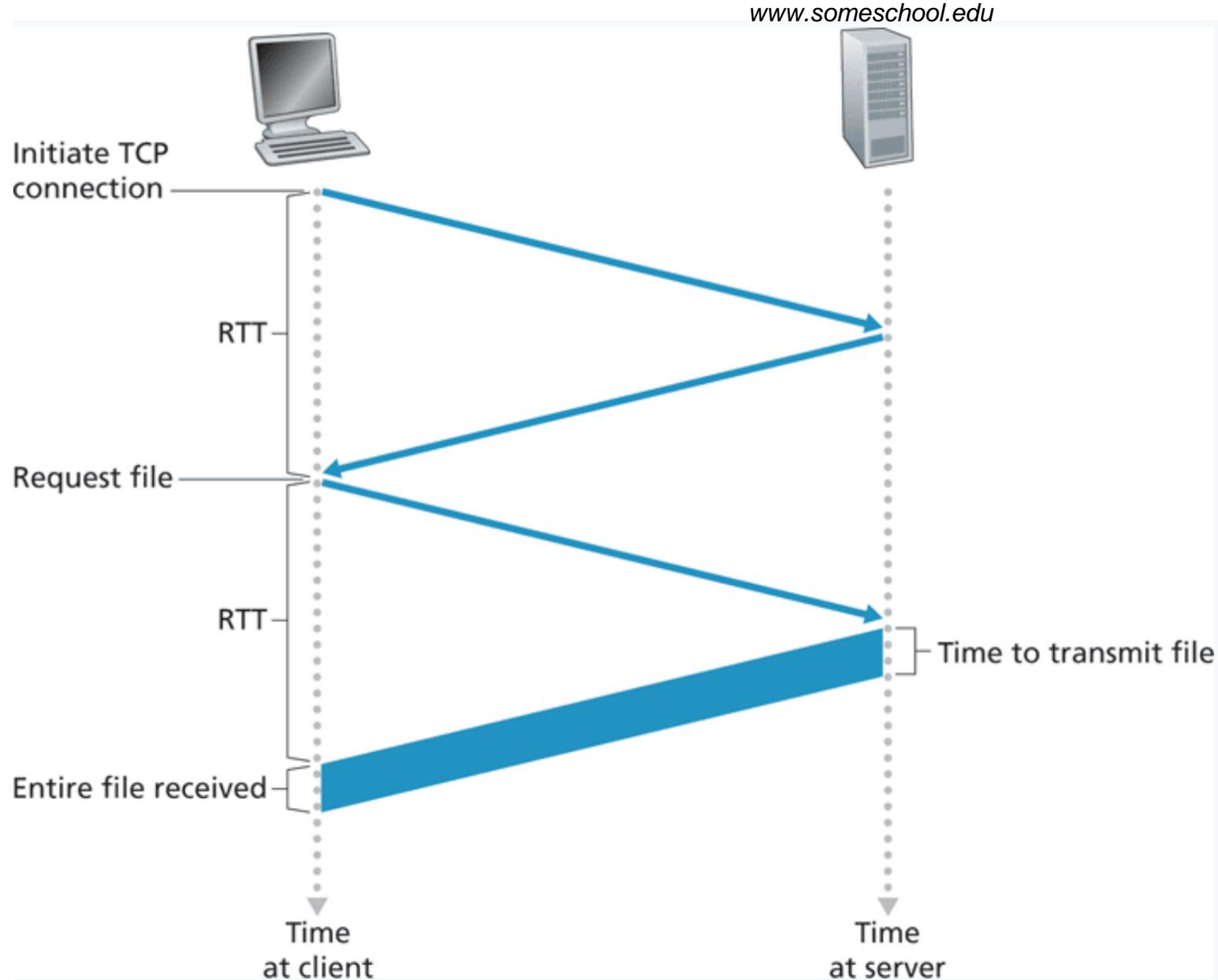
HTTP (Non Persistent) *HTTP 1.0*



1. HTTP client process initiates a TCP connection to the server on port number 80. A socket is created at both the client and the server

2. The HTTP client sends an **HTTP request** to the server via its socket. This request includes the path name for the object

HTTP (Non Persistent) *HTTP 1.0*

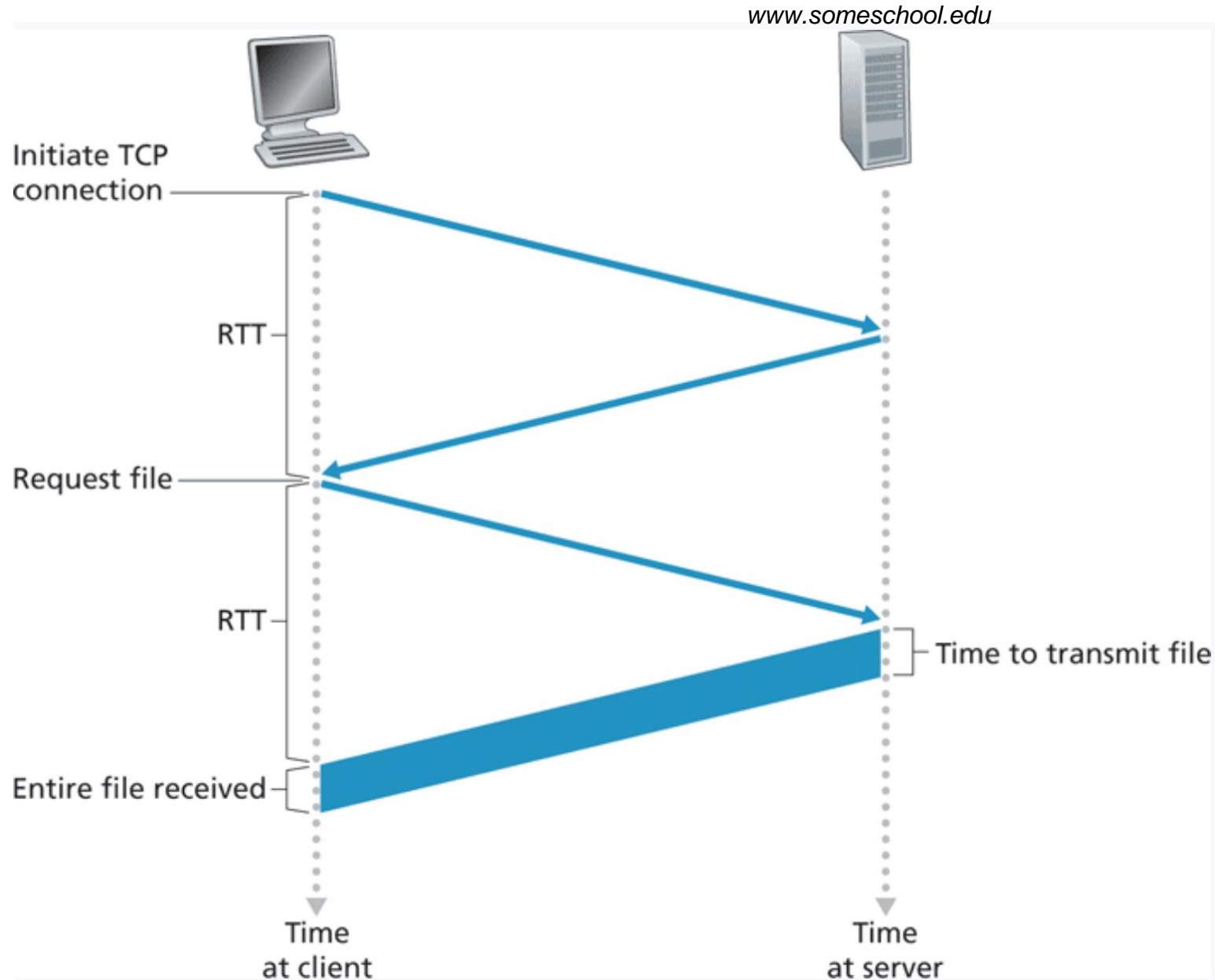


1. HTTP client process initiates a TCP connection to the server on port number 80. A socket is created at both the client and the server

2. The HTTP client sends an **HTTP request** to the server via its socket. This request includes the path name for the object

3. The HTTP server process receives the request message via its socket, retrieves the object, encapsulates the object in an **HTTP response** message, and sends the response message back to client

HTTP (Non Persistent) HTTP 1.0



1. HTTP client process initiates a TCP connection to the server on port number 80. A socket is created at both the client and the server

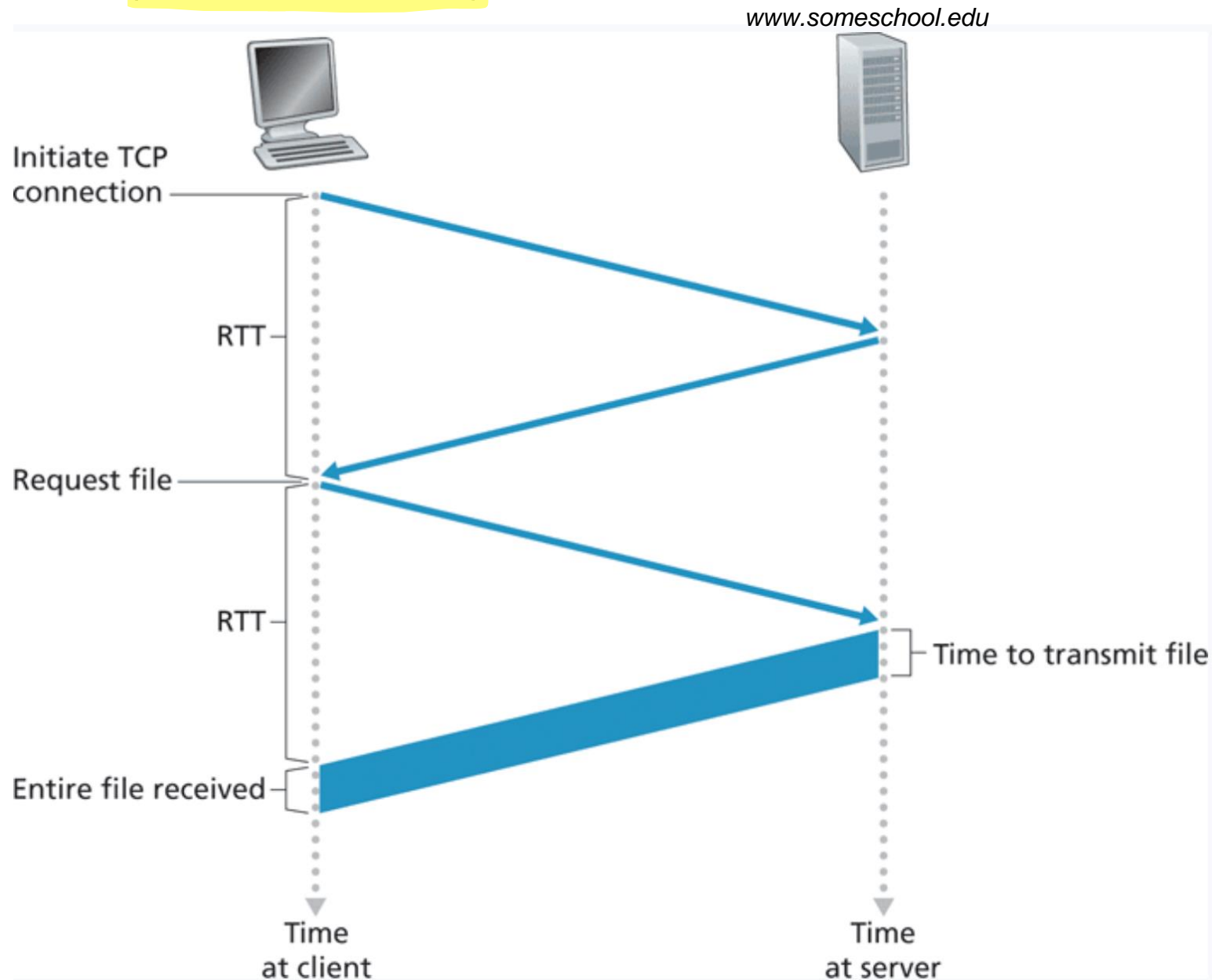
2. The HTTP client sends an **HTTP request** to the server via its socket. This request includes the path name for the object

3. The HTTP server process receives the request message via its socket, retrieves the object, encapsulates the object in an **HTTP response** message, and sends the response message back to client

4. Client receives the HTTP response. Once entire file is received, then TCP connection is closed

(repeat for each web object)

HTTP (Non Persistent) HTTP 1.0



1. HTTP client process initiates a TCP connection to the server on port number 80. A socket is created at both the client and the server

2. The HTTP client sends an **HTTP request** to the server via its socket. This request includes the path name for the object

3. The HTTP server process receives the request message via its socket, retrieves the object, encapsulates the object in an **HTTP response** message, and sends the response message back to client

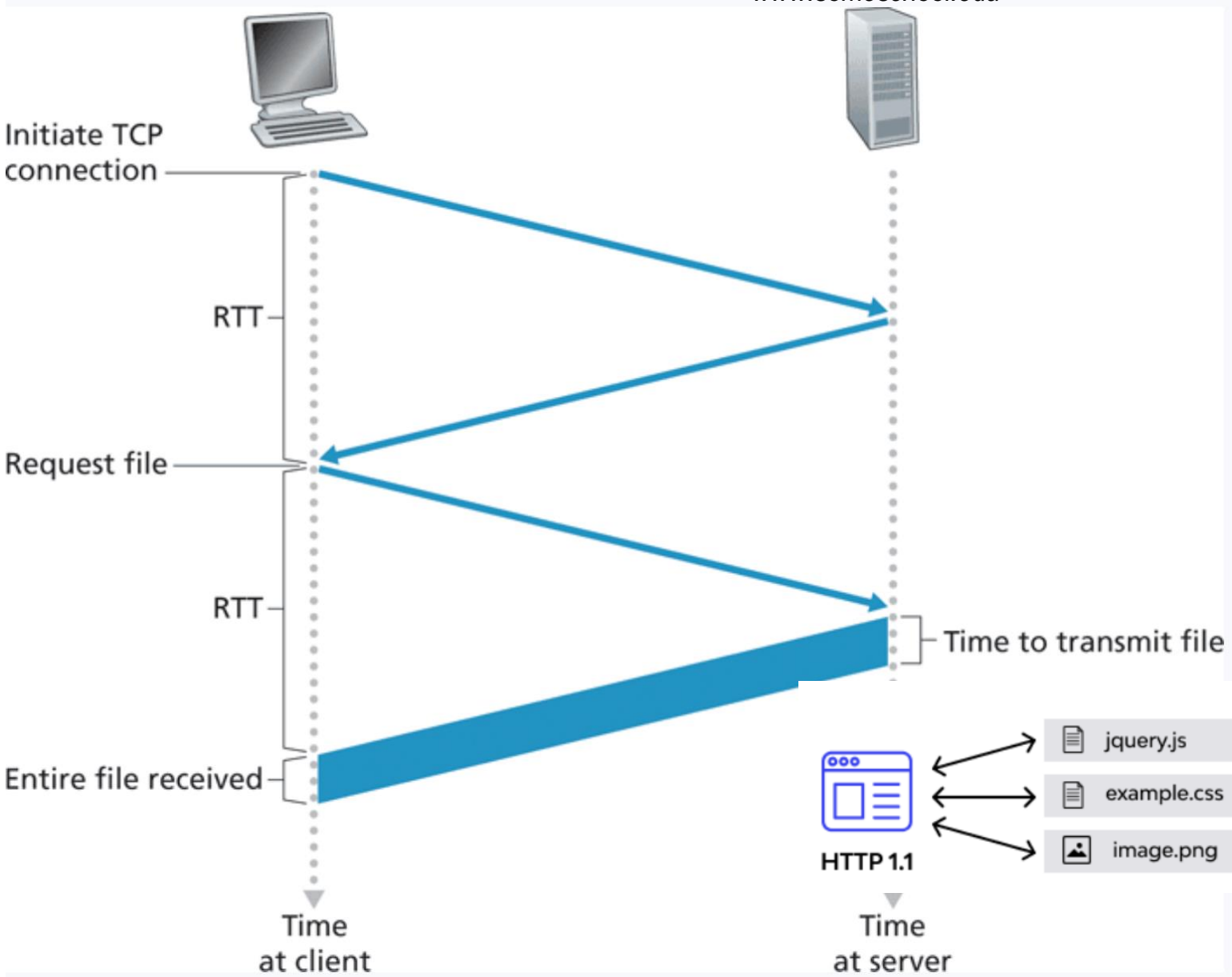
4. Client receives the HTTP response. Once entire file is received, **then TCP connection is closed**

(repeat for each web object)

HTTP (Persistent)

HTTP 1.1+

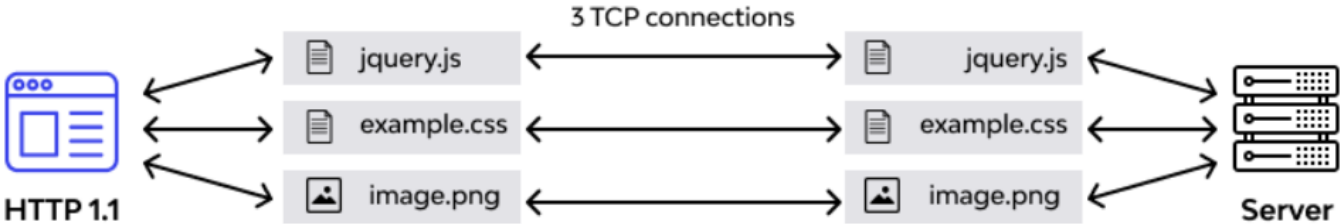
www.someschool.edu



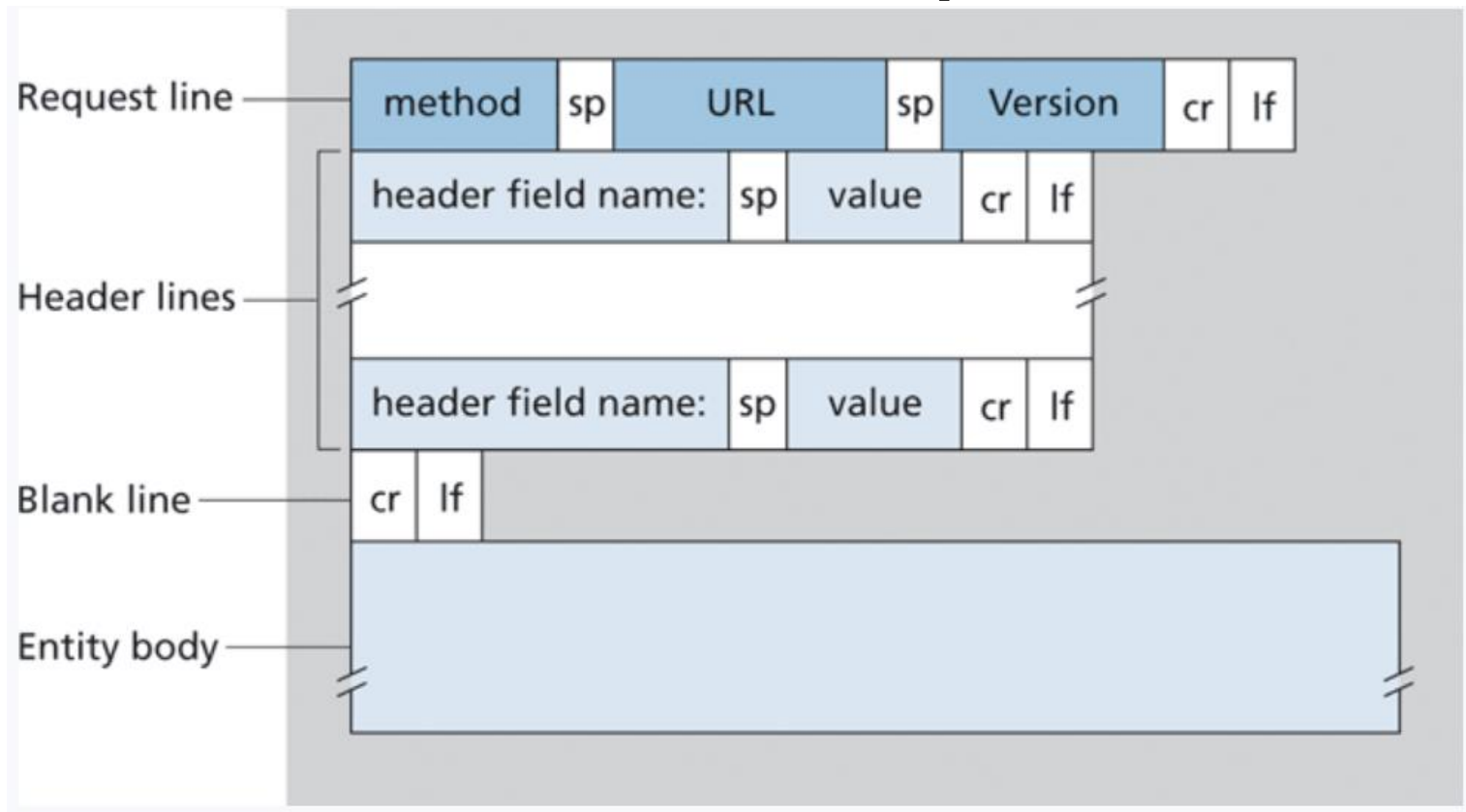
In a persistent HTTP connection, all web objects will be sent through the same TCP connection



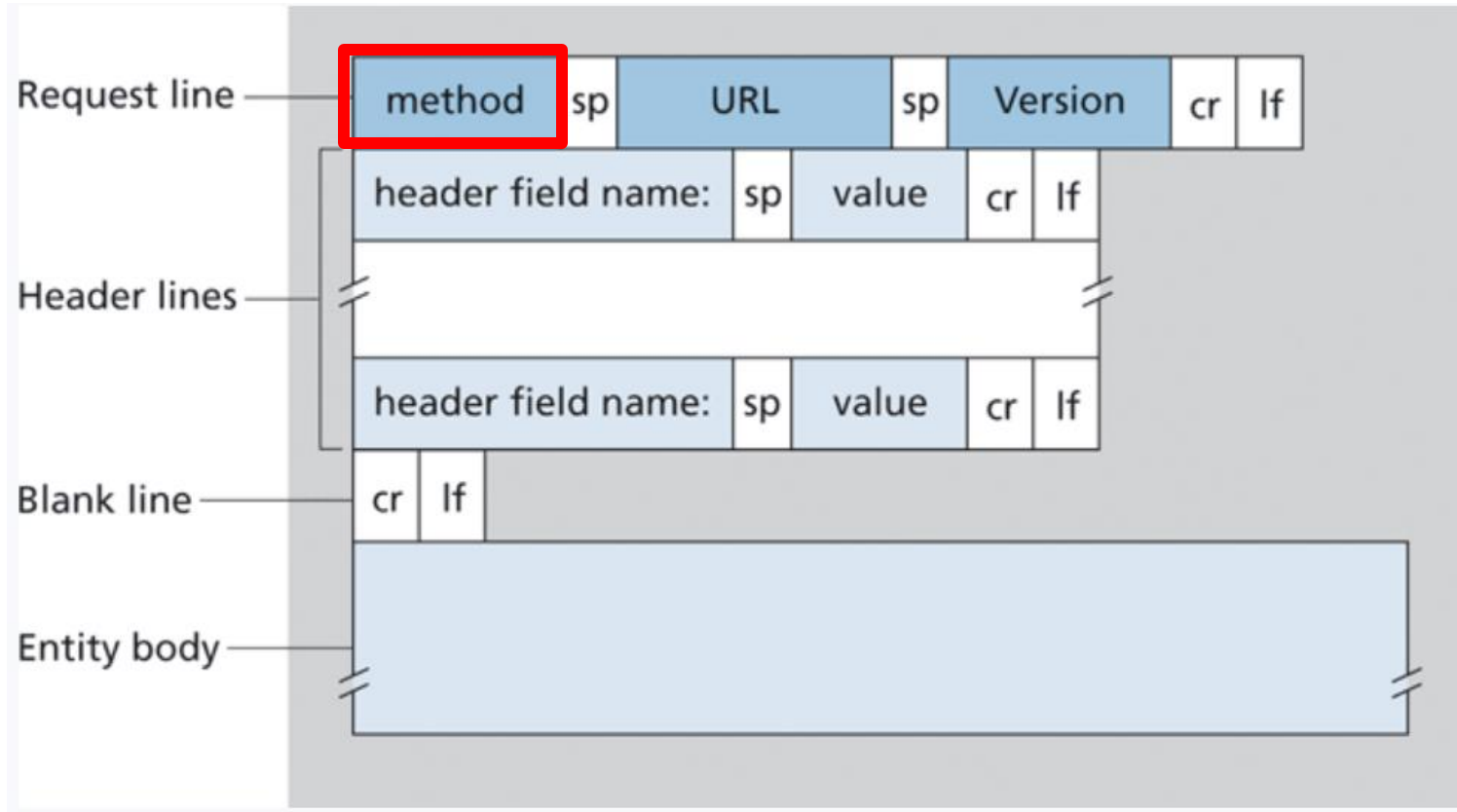
Parallelism?



HTTP Request



HTTP Request

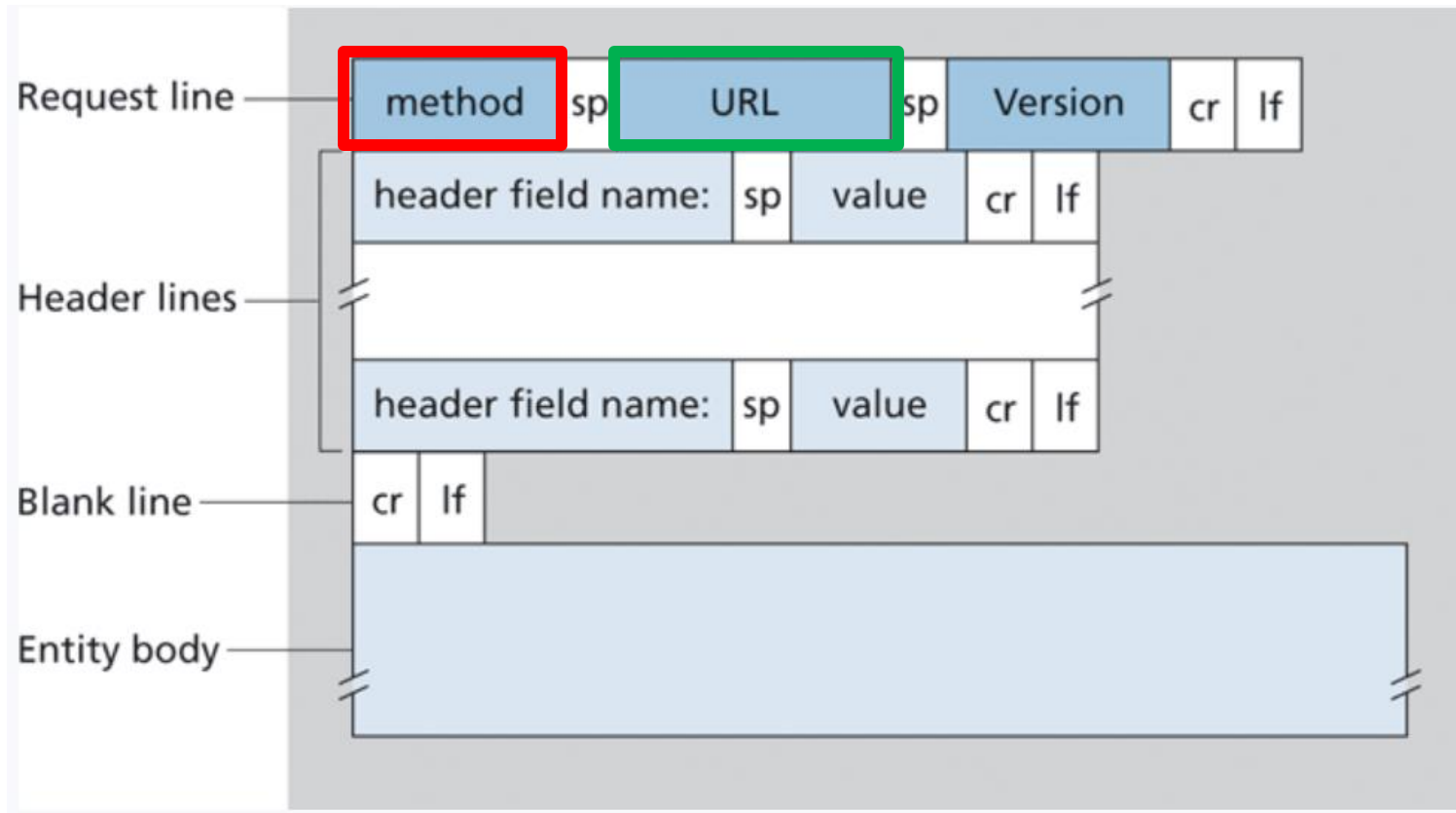


GET: Download resource
HEAD: Get resource metadata
POST: Upload form contents
PUT: Upload object to URL
DELETE: Delete object from URL

Whenever we download a web object from a web page, we issue an **HTTP GET request**



HTTP Request

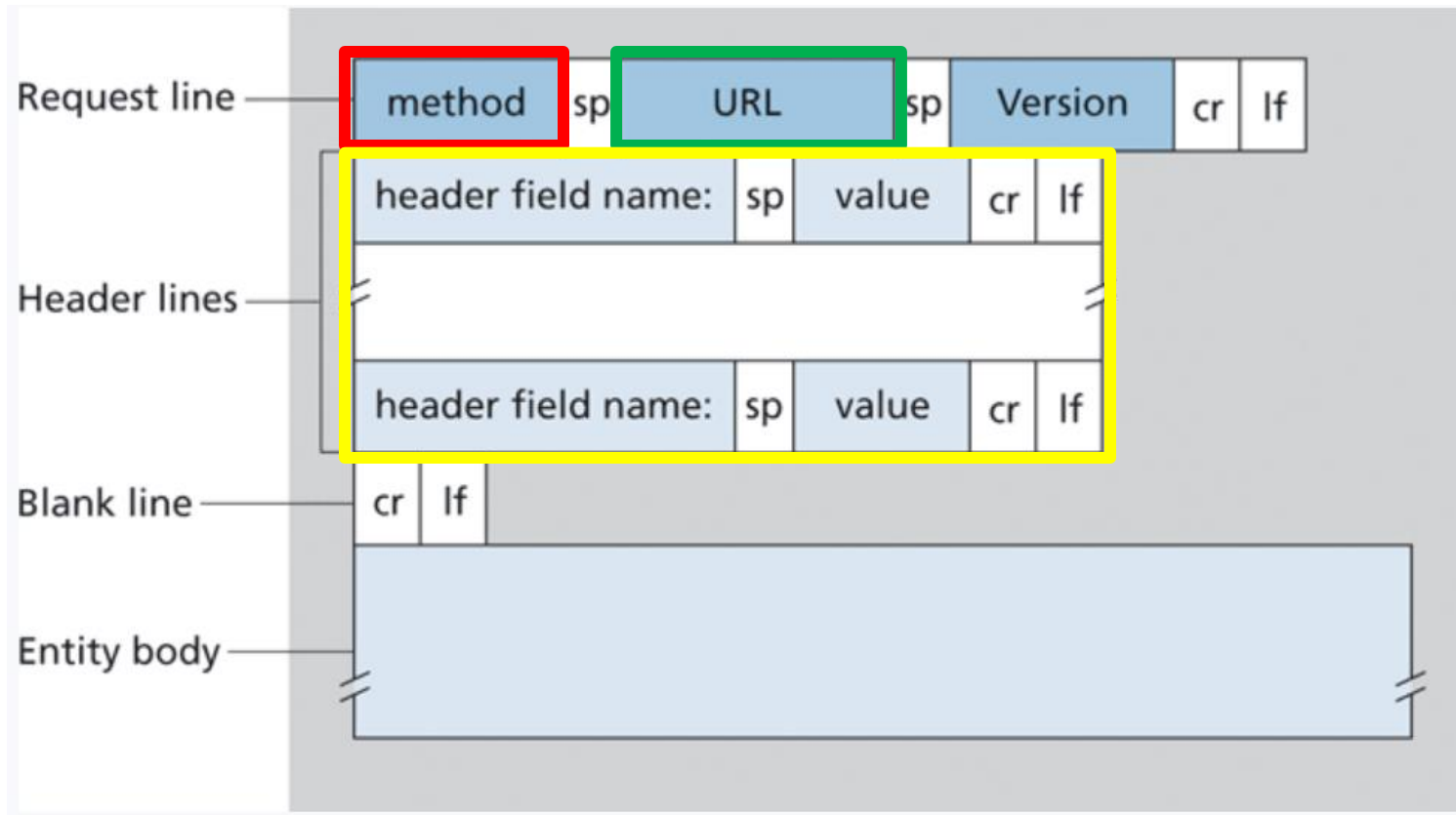


GET: Download resource
HEAD: Get resource metadata
POST: Upload form contents
PUT: Upload object to URL
DELETE: Delete object from URL

Whenever we download a web object from a web page, we issue an **HTTP GET request**

`www.someschool.edu/dog.jpeg`

HTTP Request



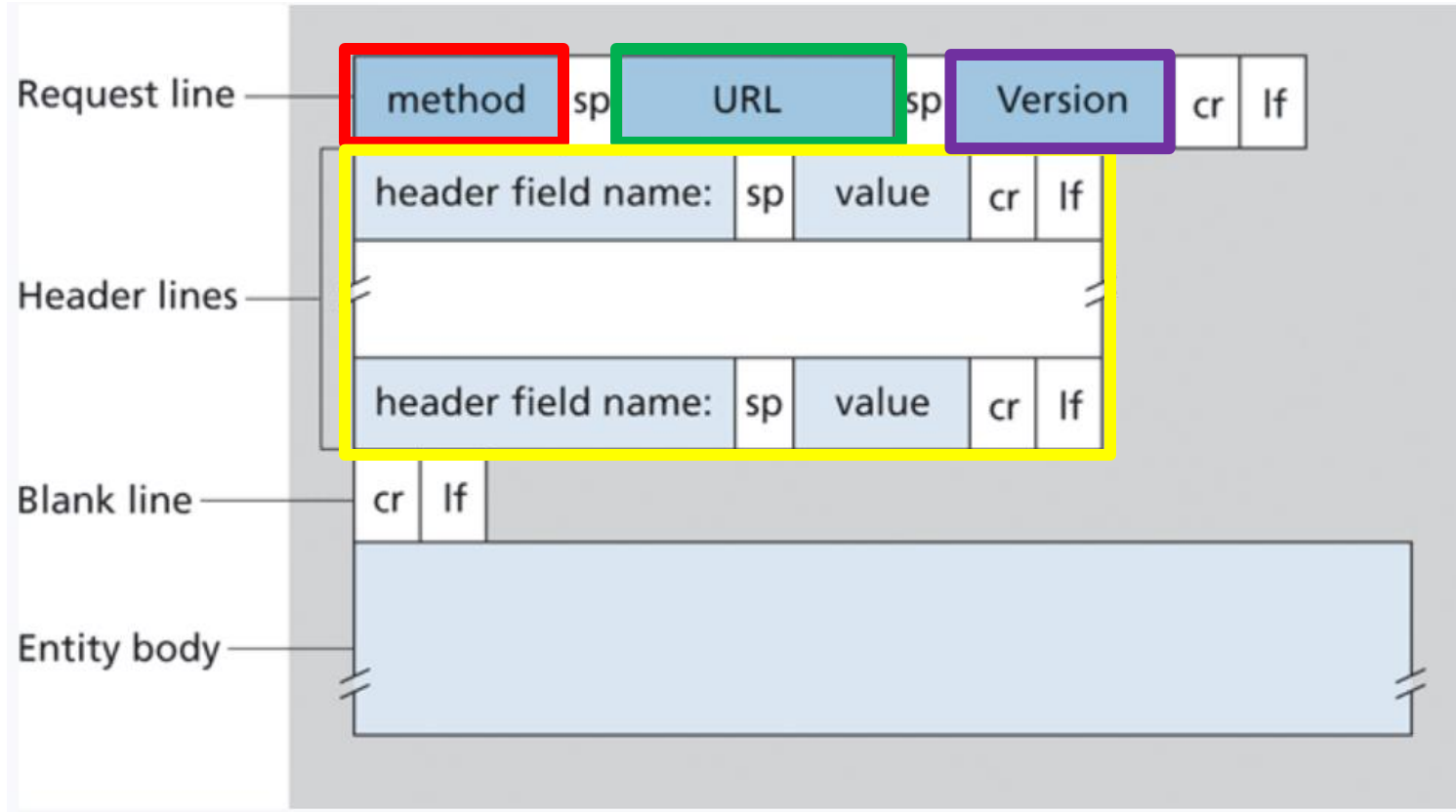
GET: Download resource
HEAD: Get resource metadata
POST: Upload form contents
PUT: Upload object to URL
DELETE: Delete object from URL

Whenever we download a web object from a web page, we issue an **HTTP GET request**

`www.someschool.edu/dog.jpeg`

```
Host: www.someschool.edu
Connection: close
User-agent: Mozilla/5.0
Accept-language: fr
```


HTTP Request



HTTP/1.1

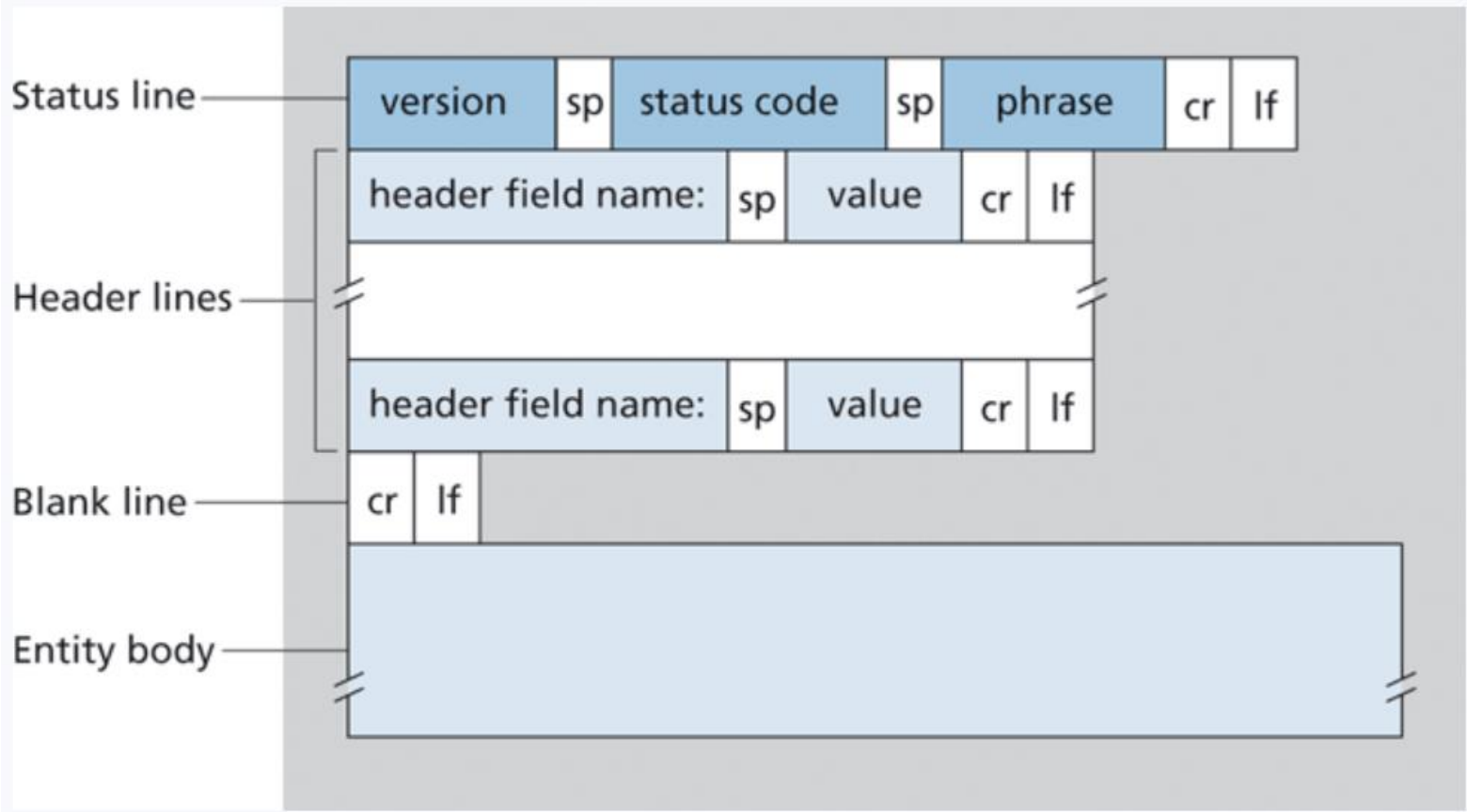
GET: Download resource
HEAD: Get resource metadata
POST: Upload form contents
PUT: Upload object to URL
DELETE: Delete object from URL

Whenever we download a web object from a web page, we issue an **HTTP GET request**

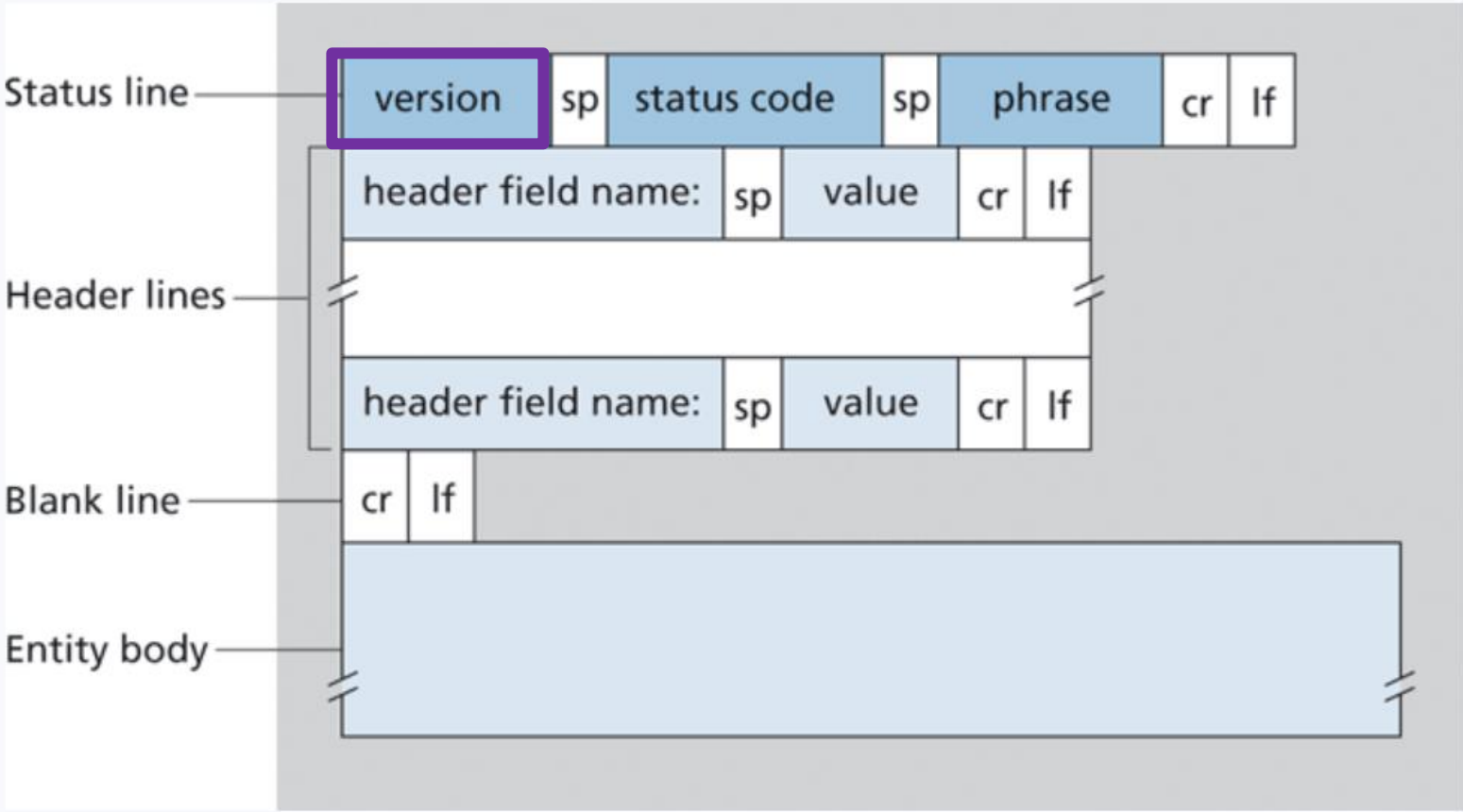
`www.someschool.edu/dog.jpeg`

```
Host: www.someschool.edu
Connection: close
User-agent: Mozilla/5.0
Accept-language: fr
```

HTTP Response

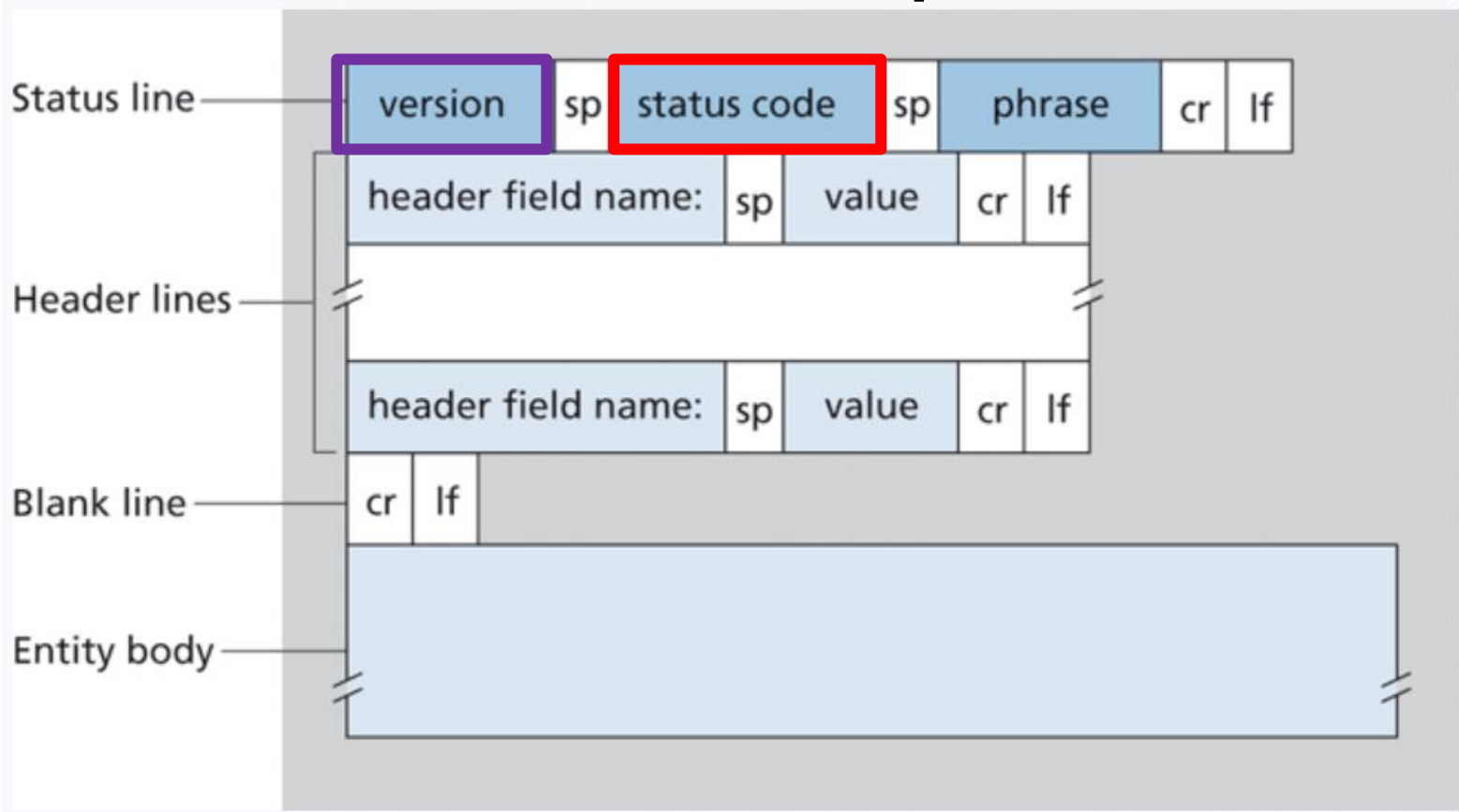


HTTP Response



HTTP/1.1

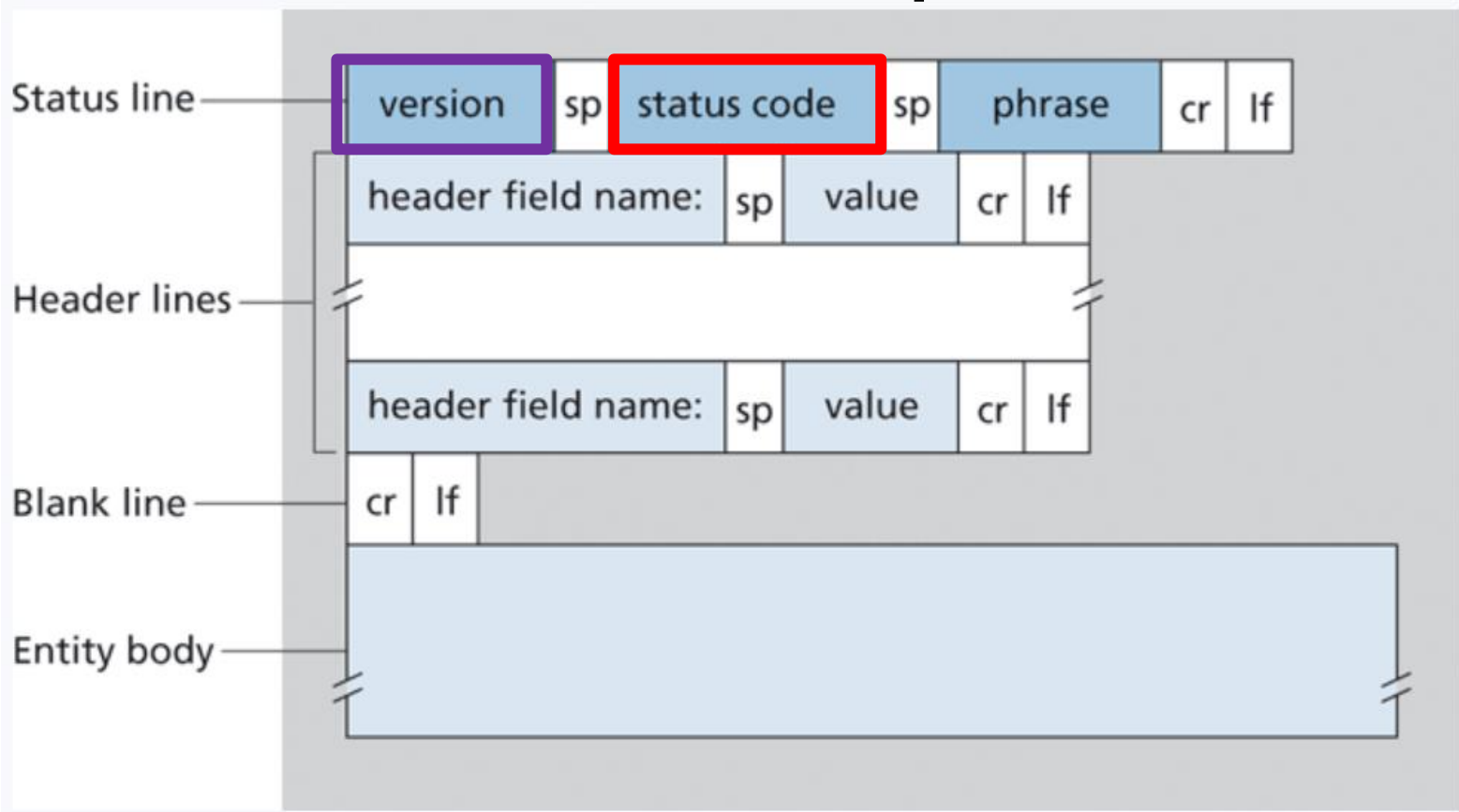
HTTP Response



200 - Ok

HTTP/1.1

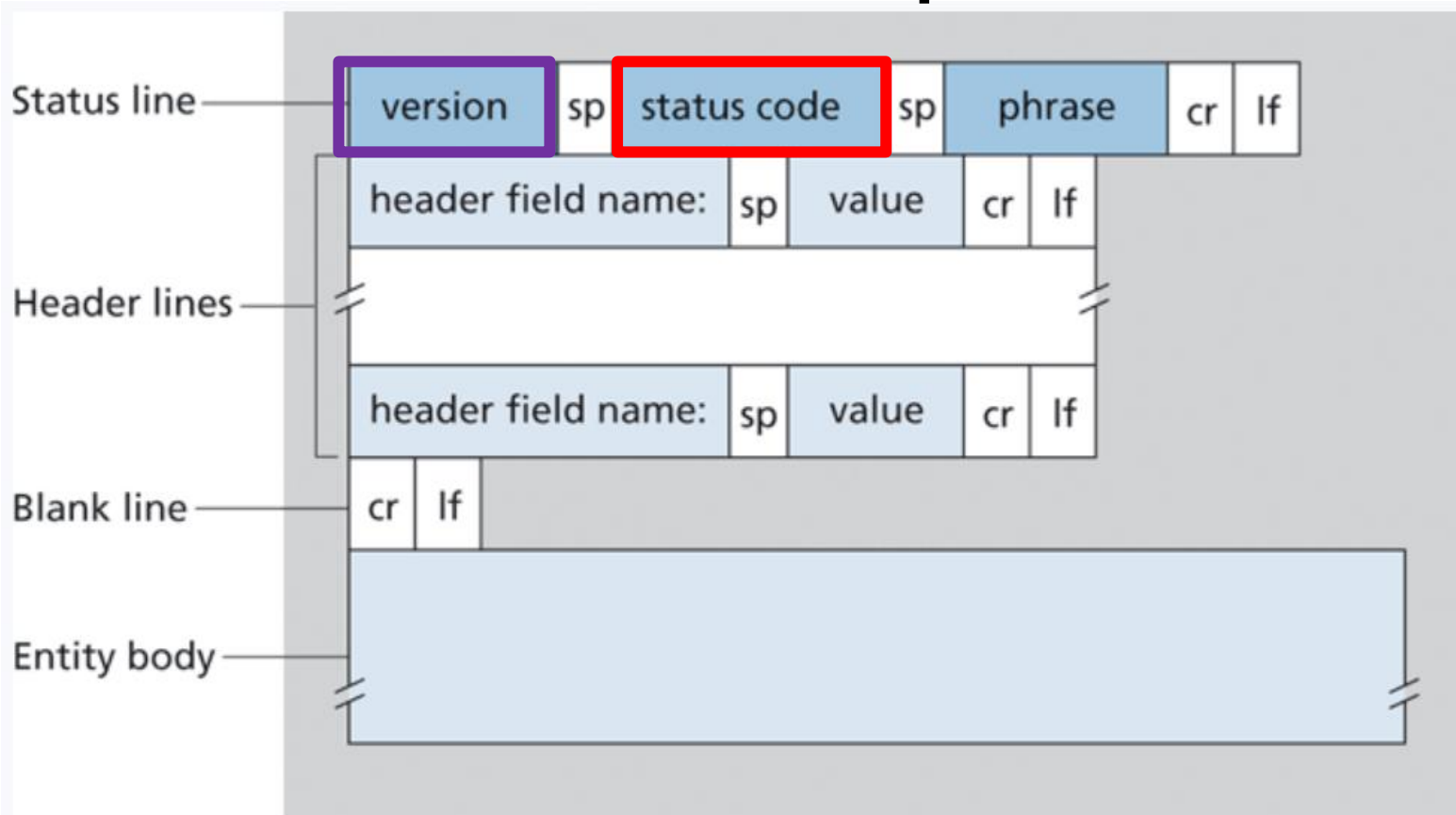
HTTP Response



200 – Ok
404 – Not found

HTTP/1.1

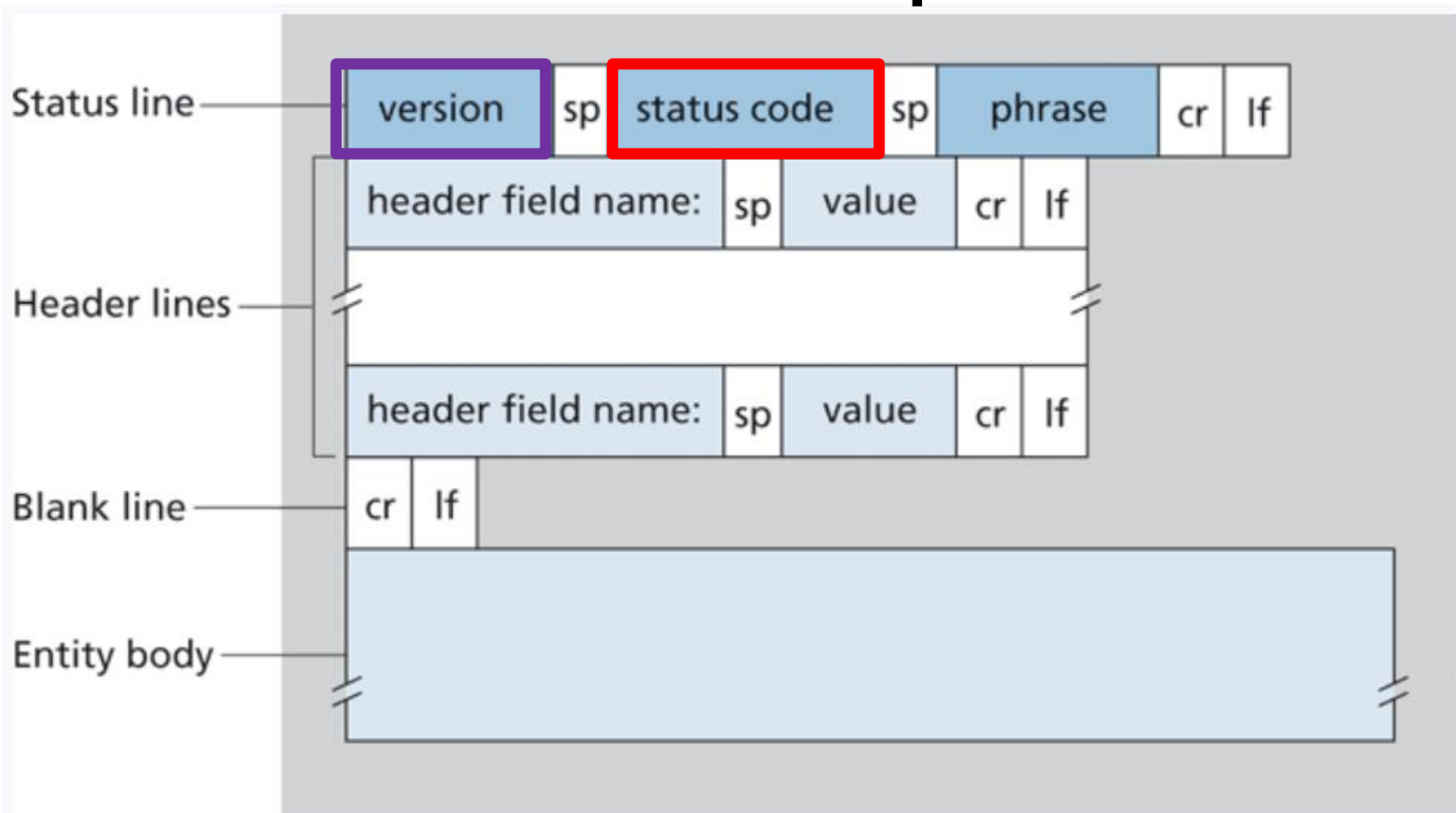
HTTP Response



HTTP/1.1

200 – Ok
404 – Not found
301 – Resource
has moved

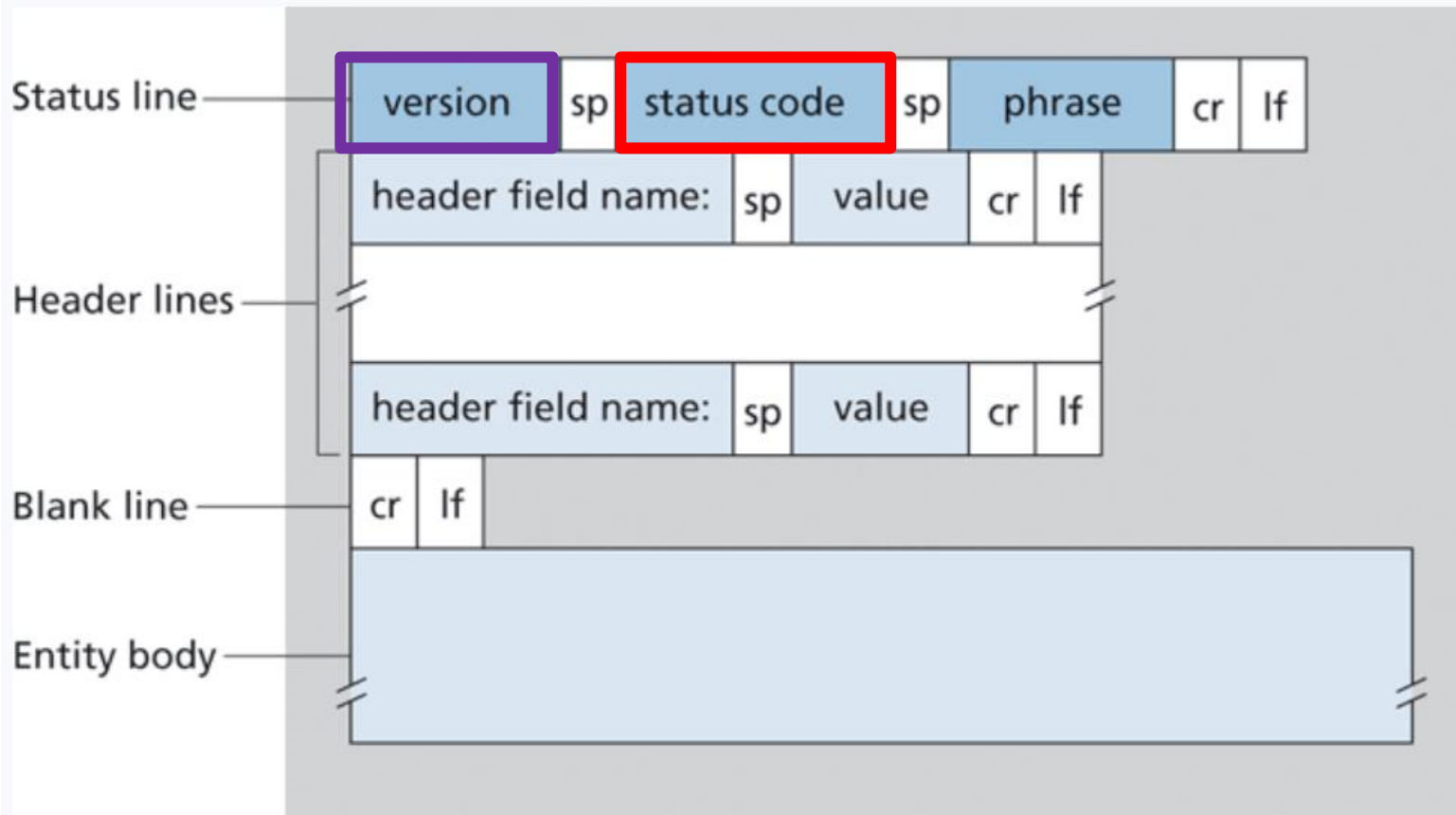
HTTP Response



HTTP/1.1

200 – Ok
404 – Not found
301 – Resource
has moved
500 – Internal
Server Error

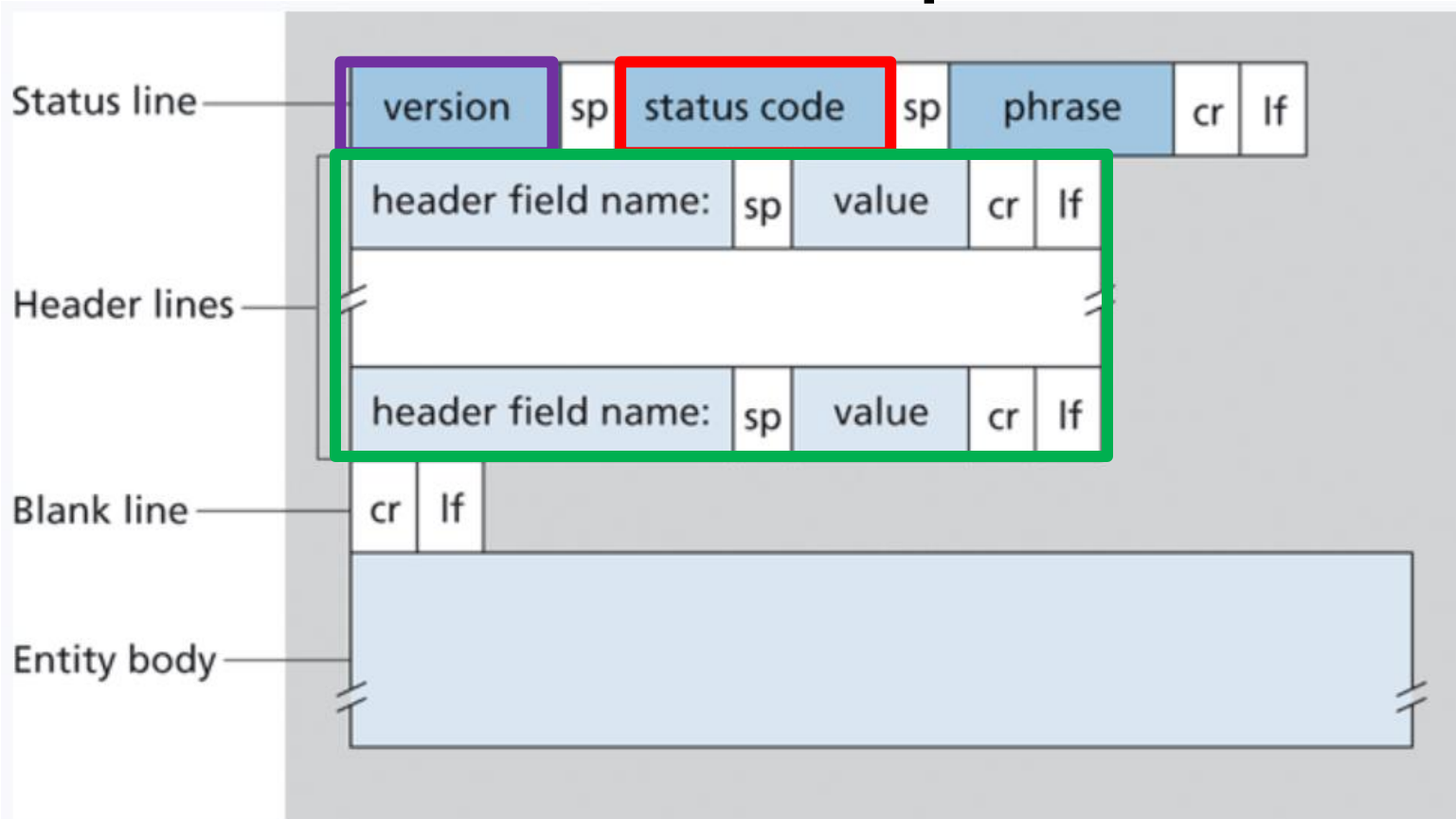
HTTP Response



- Informational Responses (100s)
- Successful Responses (200s)
- Redirection messages (300s)
- Client error response (400s)
- Server error response (500s)

HTTP/1.1

HTTP Response

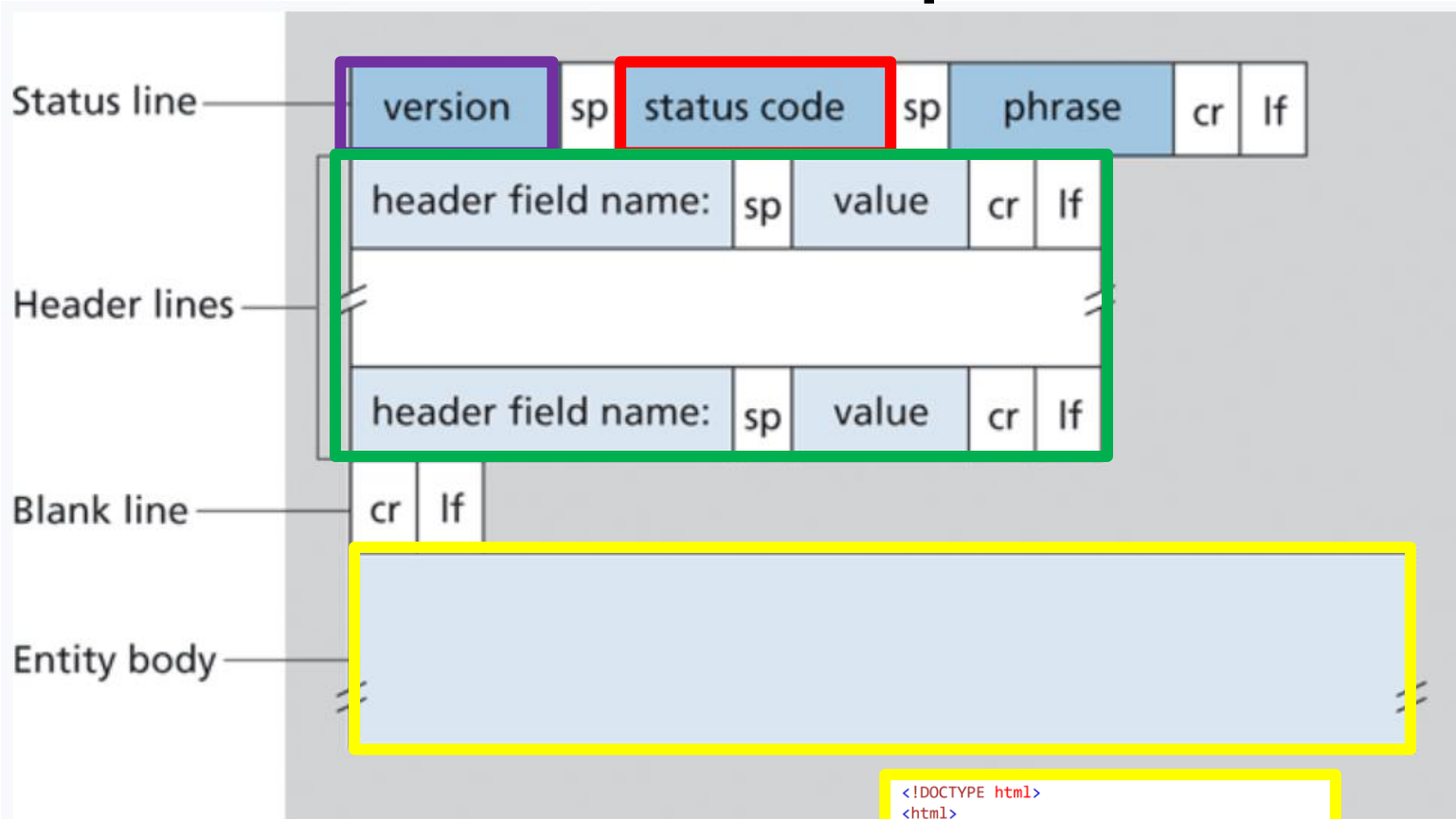


- Informational Responses (100s)
- Successful Responses (200s)
- Redirection messages (300s)
- Client error response (400s)
- Server error response (500s)

```
HTTP/1.1 200 OK
Connection: close
Date: Tue, 18 Aug 2015 15:44:04 GMT
Server: Apache/2.2.3 (CentOS)
Last-Modified: Tue, 18 Aug 2015 15:11:03 GMT
Content-Length: 6821
Content-Type: text/html
```

HTTP/1.1

HTTP Response



- Informational Responses (100s)
- Successful Responses (200s)
- Redirection messages (300s)
- Client error response (400s)
- Server error response (500s)

```
HTTP/1.1 200 OK
Connection: close
Date: Tue, 18 Aug 2015 15:44:04 GMT
Server: Apache/2.2.3 (CentOS)
Last-Modified: Tue, 18 Aug 2015 15:11:03 GMT
Content-Length: 6821
Content-Type: text/html
```

HTTP/1.1

```
<!DOCTYPE html>
<html>
<head>
</head>
<body>

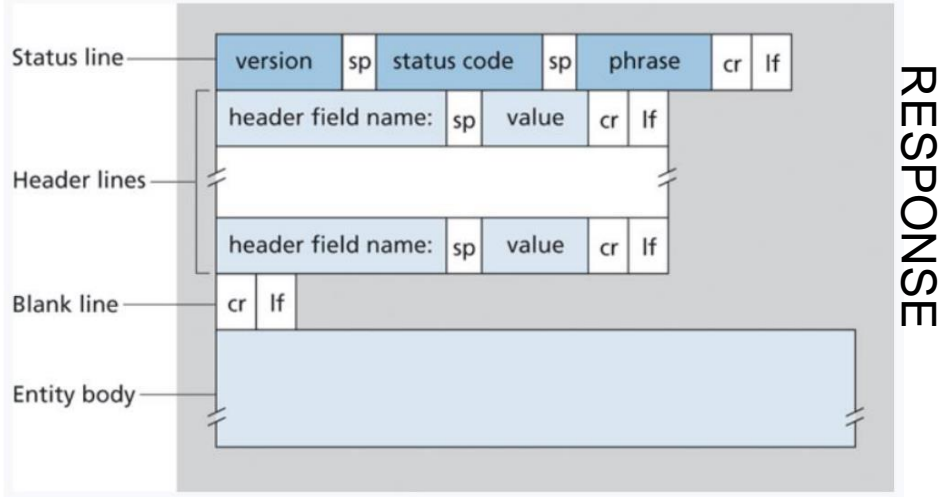
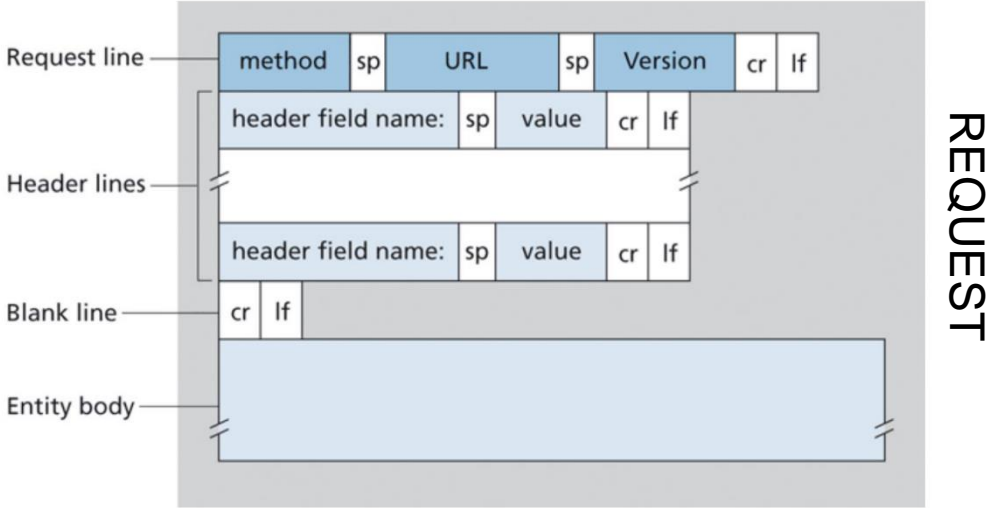
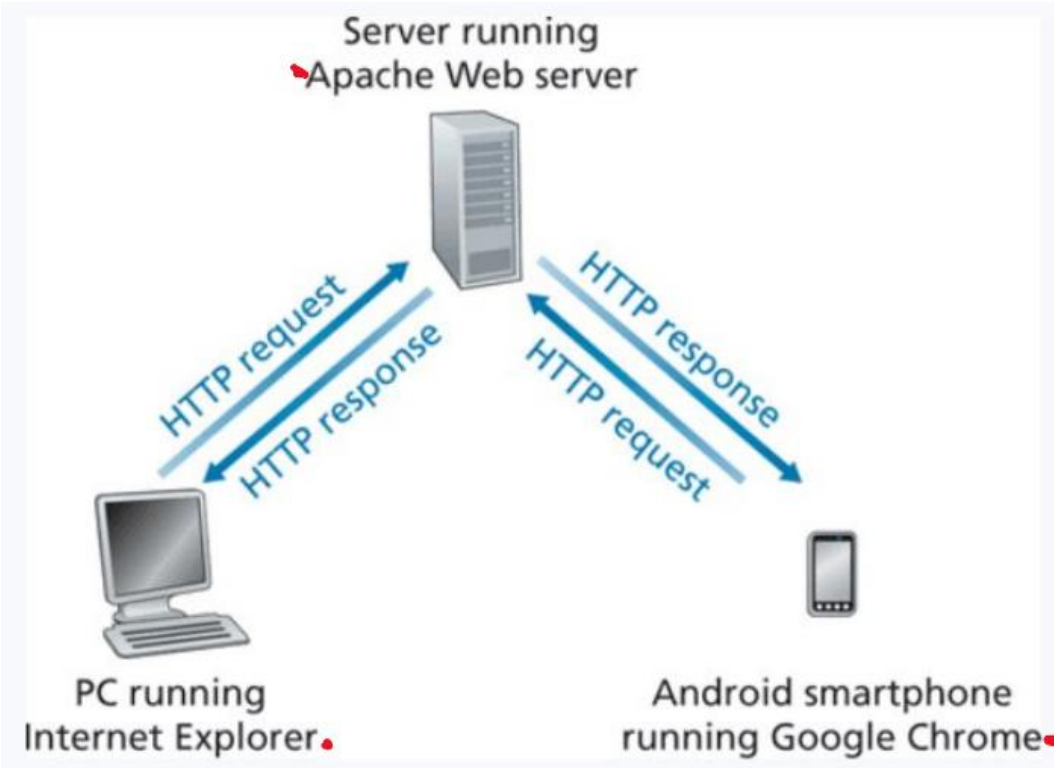
<div class="city">
  <h2>London</h2>
  <p>London is the capital of England.</p>
</div>

<div class="city">
  <h2>Paris</h2>
  <p>Paris is the capital of France.</p>
</div>
```

HTTP tl;dr

HyperText Transfer Protocol (HTTP)- protocol that dictates the transmitting of hypermedia documents such as HTML and other webpage objects

All standard web navigation is done through HTTP



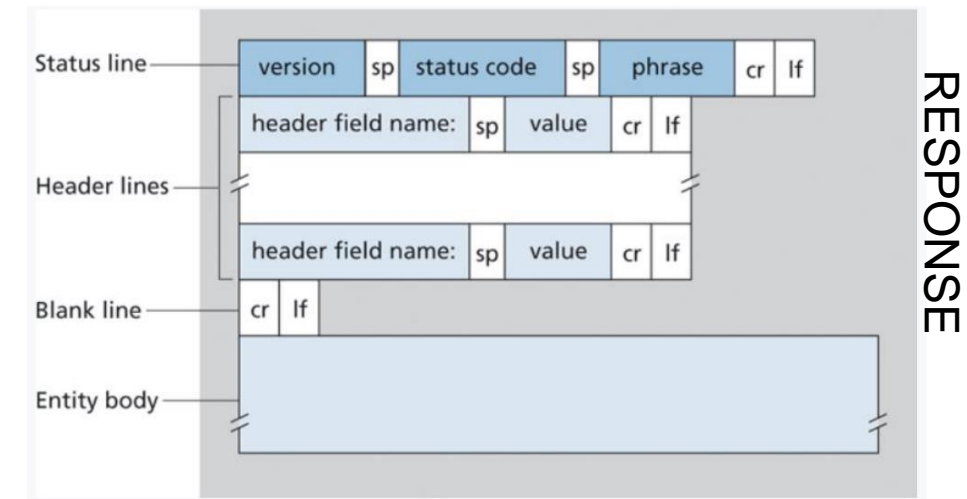
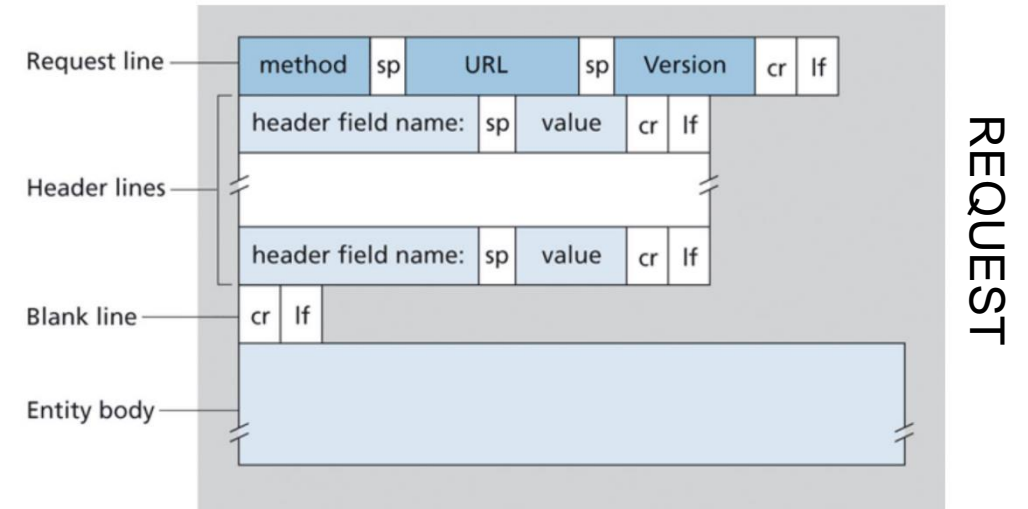
HTTP tl;dr

HyperText Transfer Protocol (HTTP)- protocol that dictates the transmitting of hypermedia documents such as HTML and other webpage objects

All standard web navigation is done through HTTP

HTTPS- is the secure implementation of HTTP

→ Adds **encryption** and **authentication**

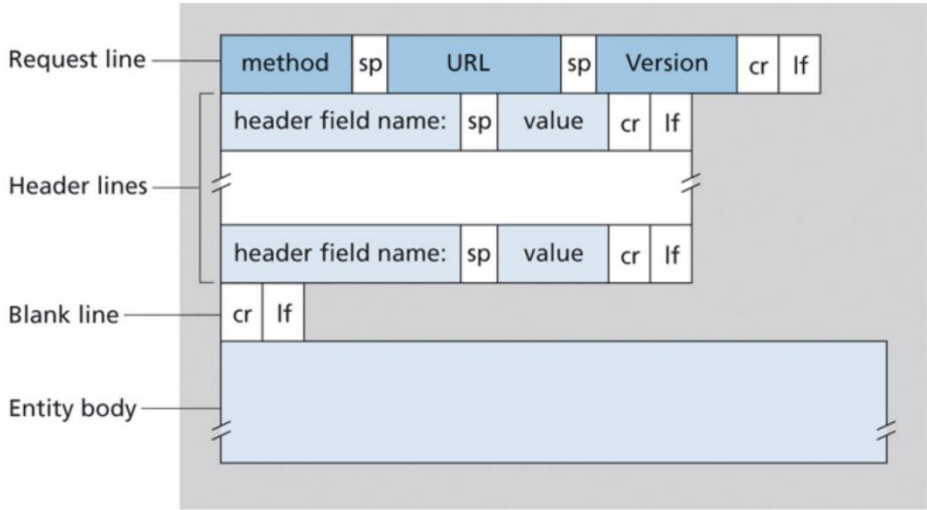


HTTP tl;dr

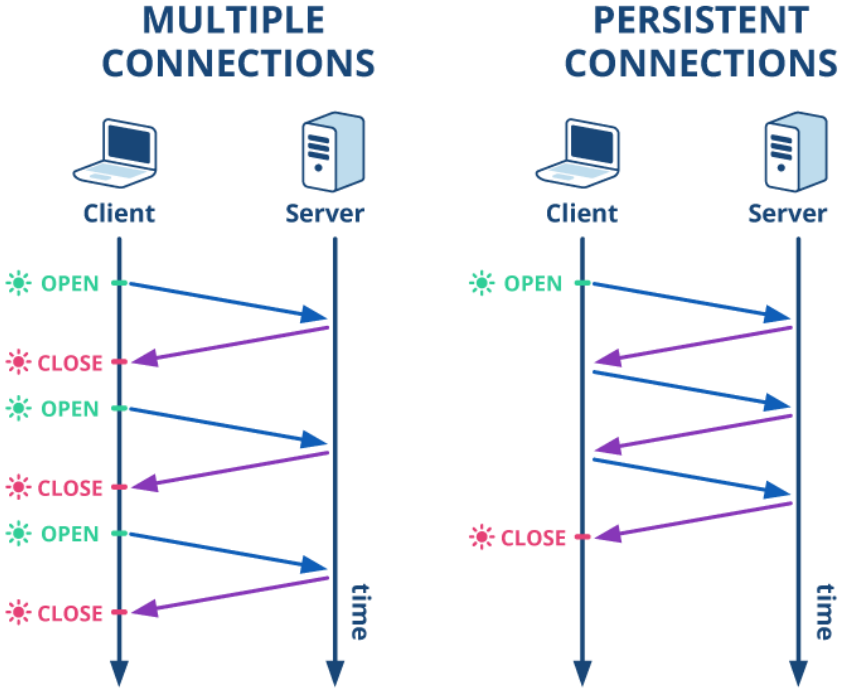
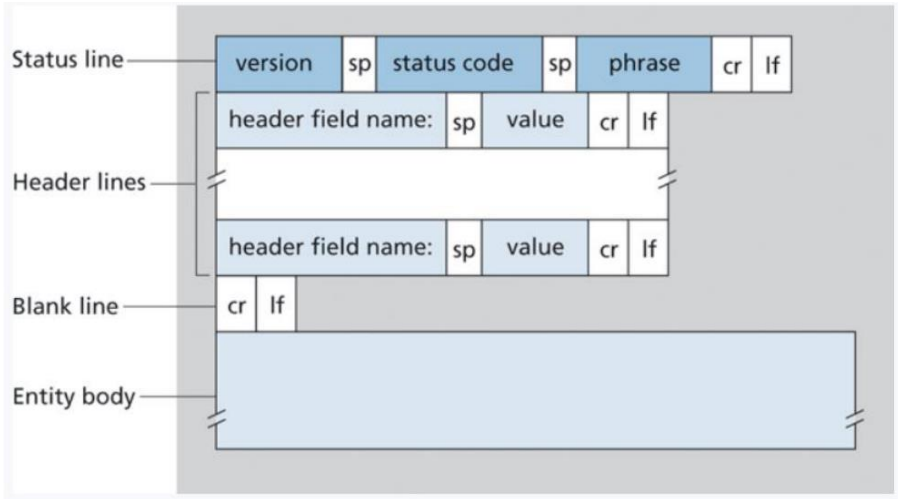
HyperText Transfer Protocol (HTTP)- protocol that dictates the transmitting of hypermedia documents such as HTML and other webpage objects

All standard web navigation is done through HTTP

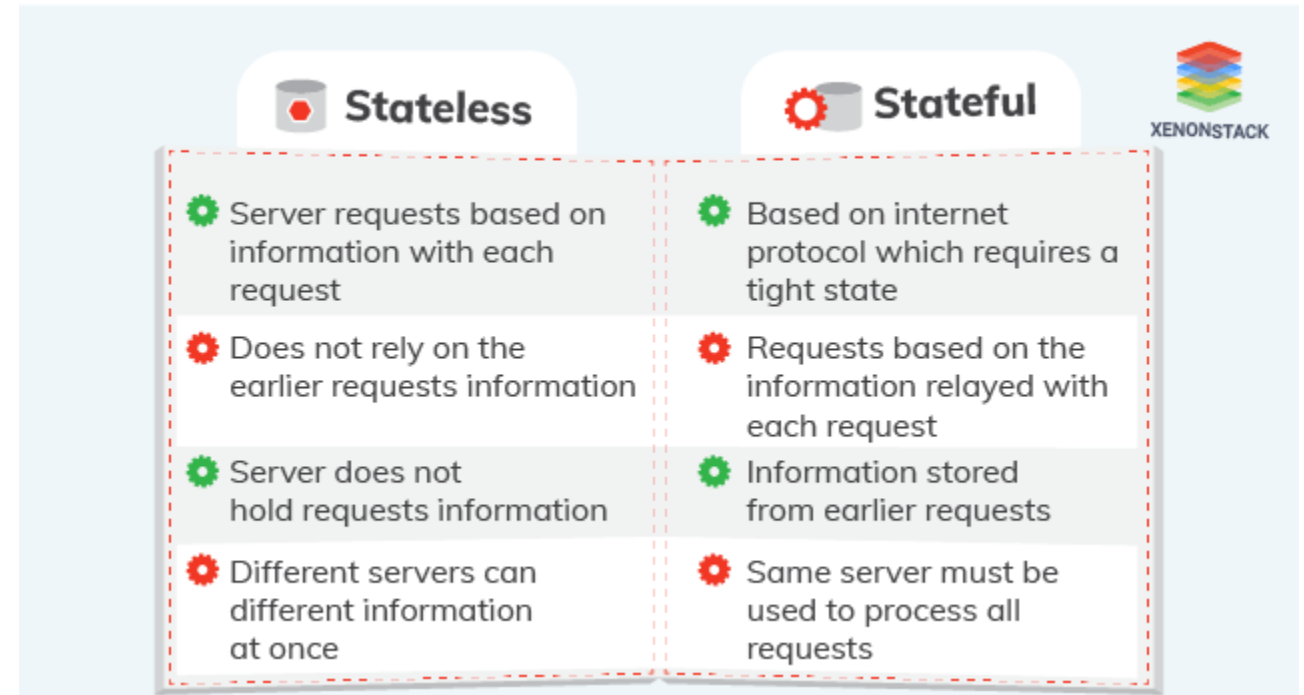
REQUEST



RESPONSE



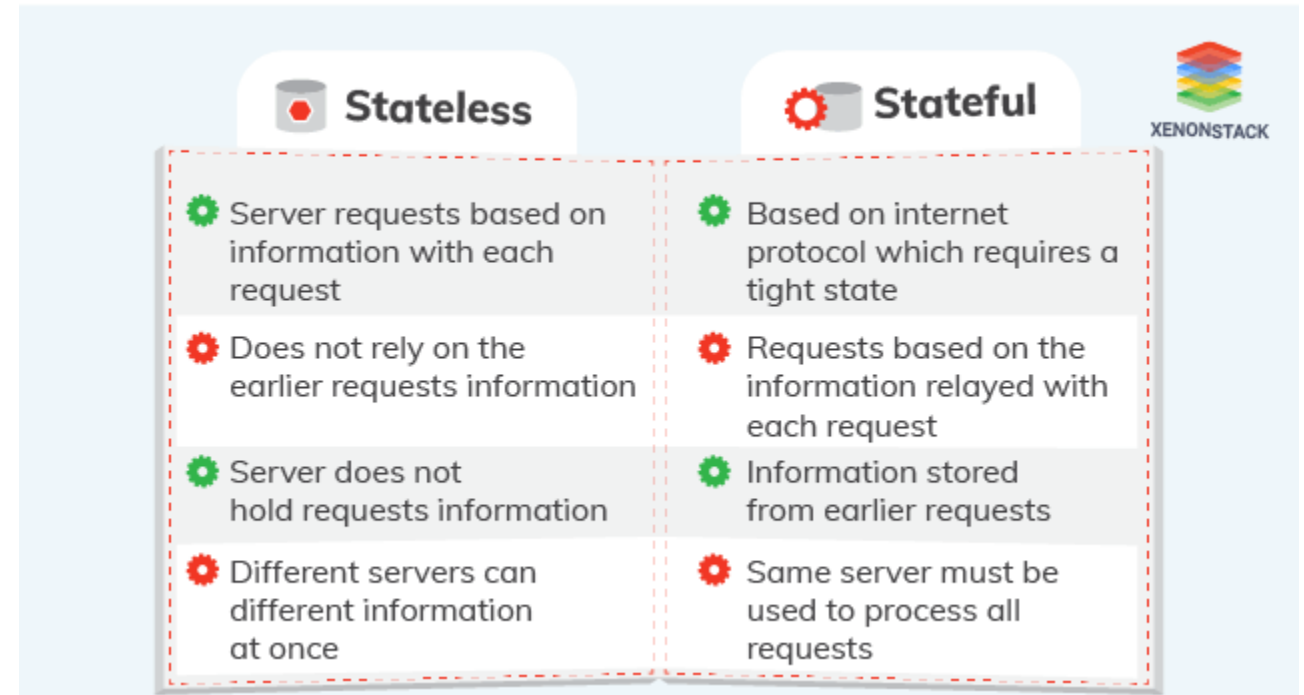
HTTP is a **stateless** protocol; it does not save information about the sender-receiver session and treats each user as a brand-new connection



HTTP is a **stateless** protocol; it does not save information about the sender-receiver session and treats each user as a brand-new connection

often it can be useful to identify a user.

- User access and permissions
- Dynamic content



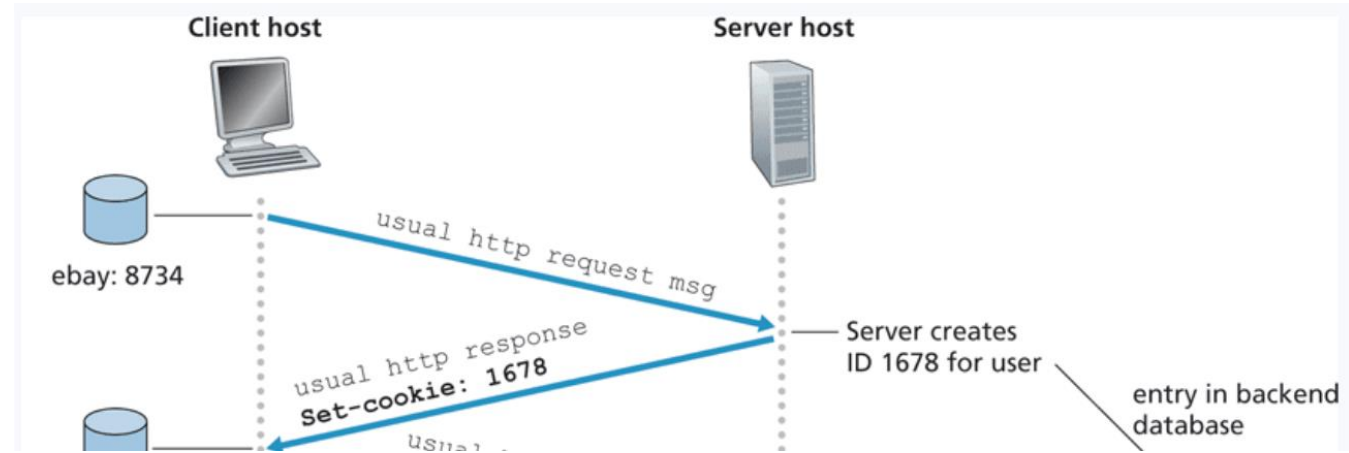
Cookies are pieces of information that are exchanged between browsers and web servers to identify users in active connections

- Authentication
- Tracking & Advertisement
- Session Management



Cookies in HTTP

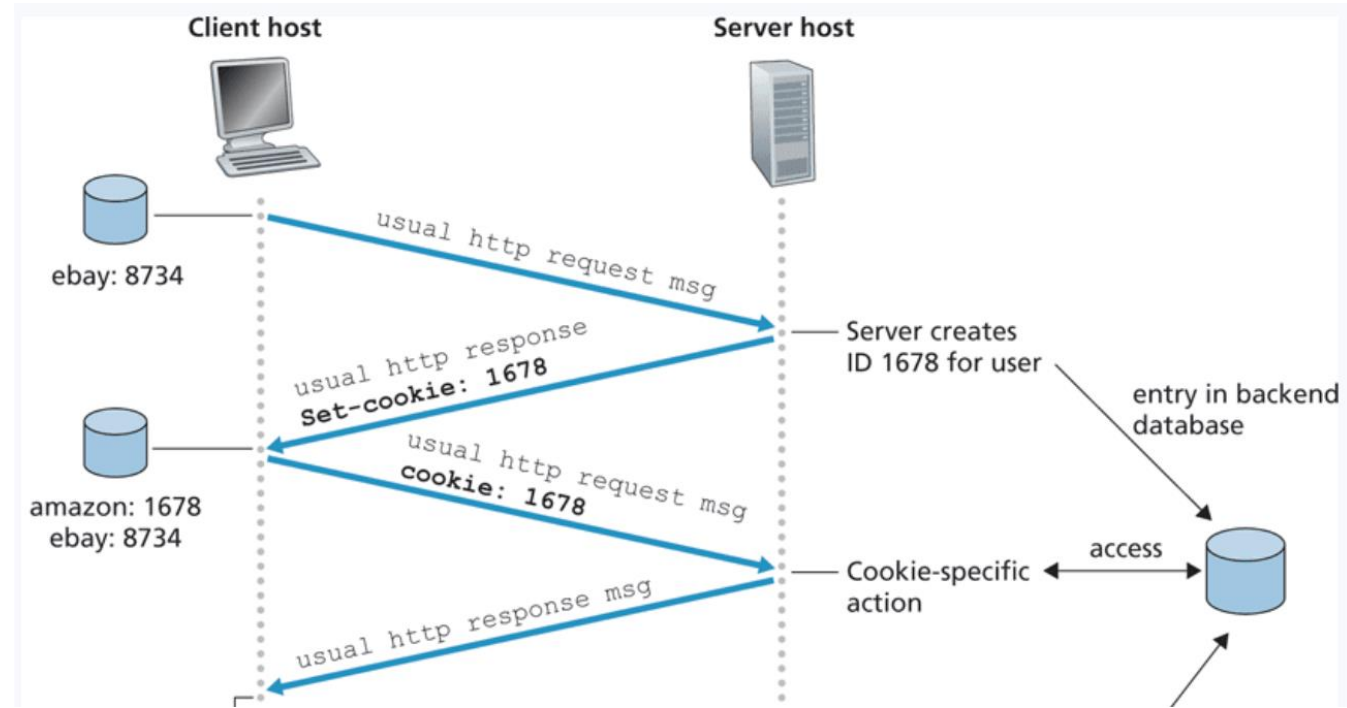
After sending a request to a webserver for the first time, the user is assigned a unique cookie value



Cookies in HTTP

After sending a request to a webserver for the first time, the user is assigned a unique cookie value

Cookie 1678 is created and stored in the user's browser as well as some database backend on the server side

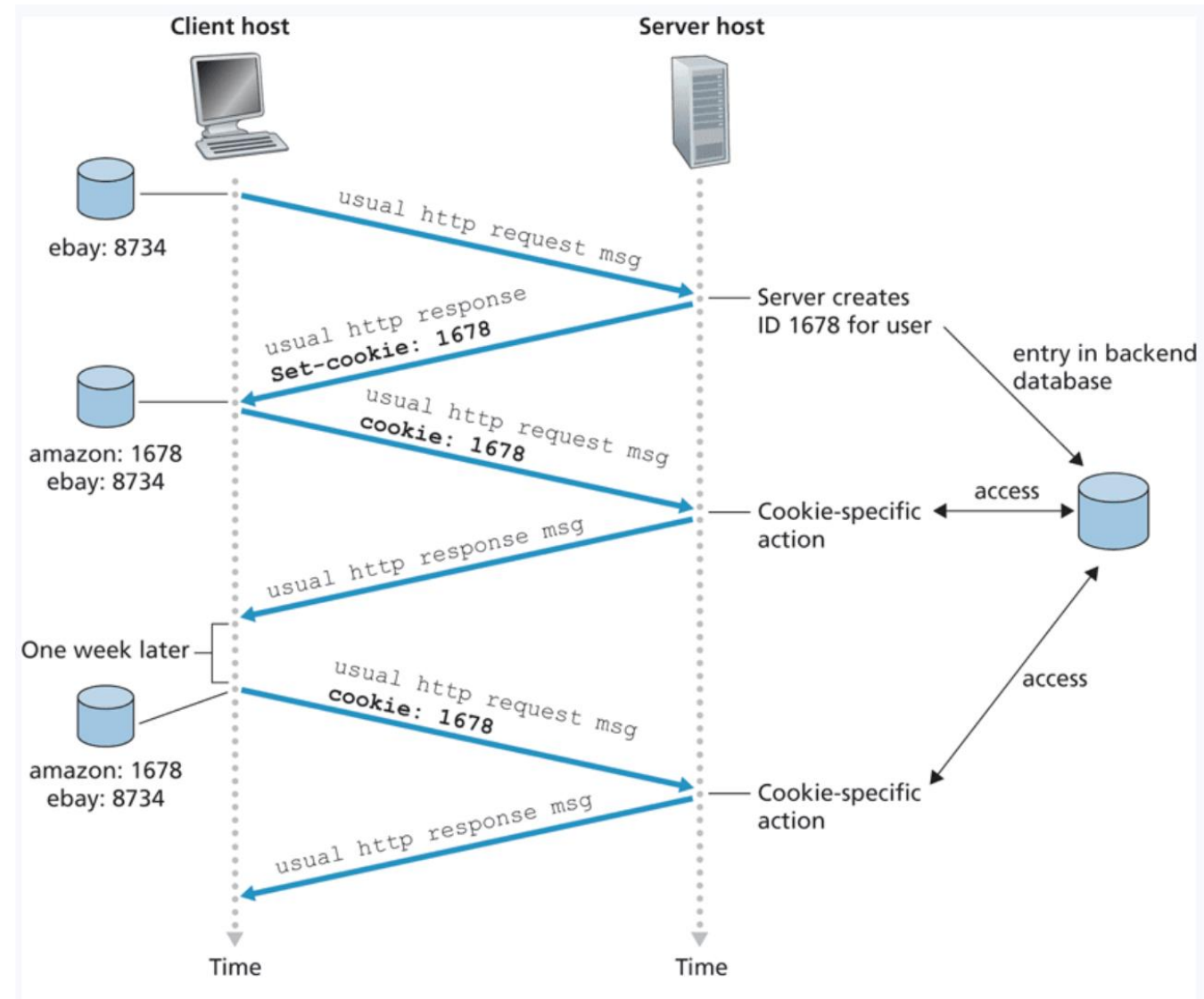


Cookies in HTTP

After sending a request to a webserver for the first time, the user is assigned a unique cookie value

Cookie 1678 is created and stored in the user's browser as well as some database backend on the server side

When the user goes back to visit the website, the cookie will be exchanged between client and host so the website can execute cookie-specific actions

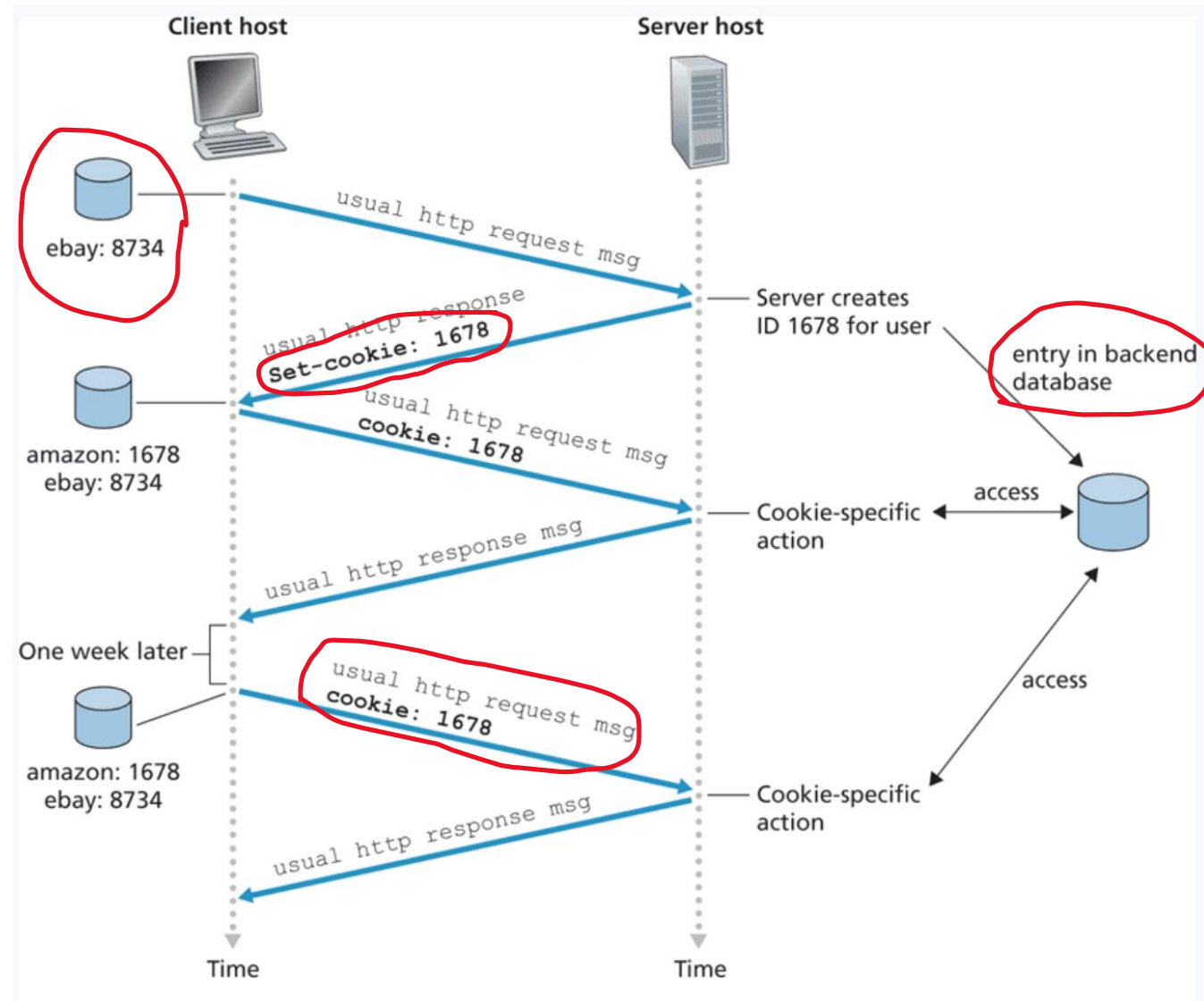


Cookies in HTTP

After sending a request to a webserver for the first time, the user is assigned a unique cookie value

Cookie 1678 is created and stored in the user's browser as well as some database backend on the server side

When the user goes back to visit the website, the cookie will be exchanged between client and host so the website can execute cookie-specific actions

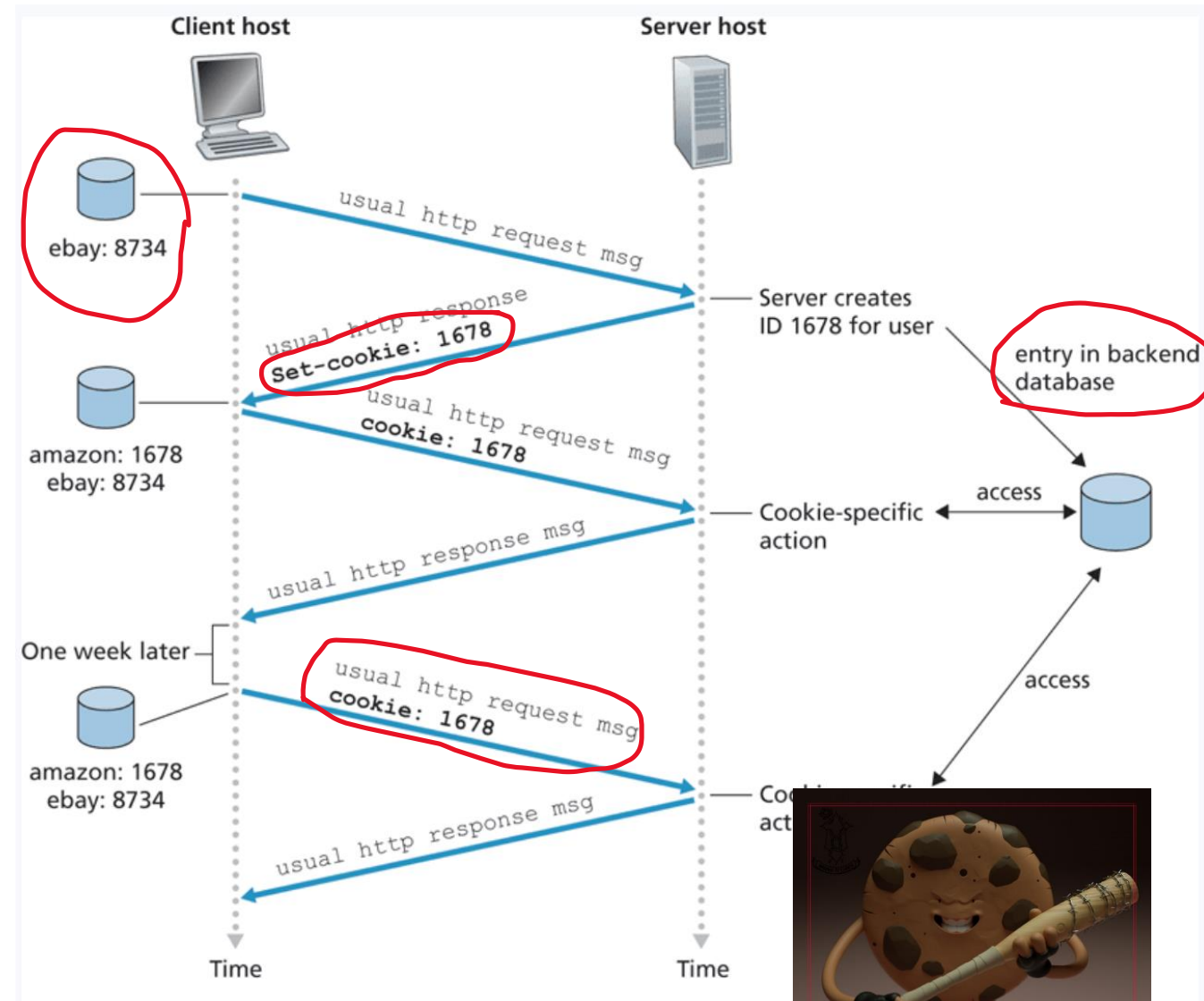


Cookies in HTTP

After sending a request to a webserver for the first time, the user is assigned a unique cookie value

Cookie 1678 is created and stored in the user's browser as well as some database backend on the server side

When the user goes back to visit the website, the cookie will be exchanged between client and host so the website can execute cookie-specific actions

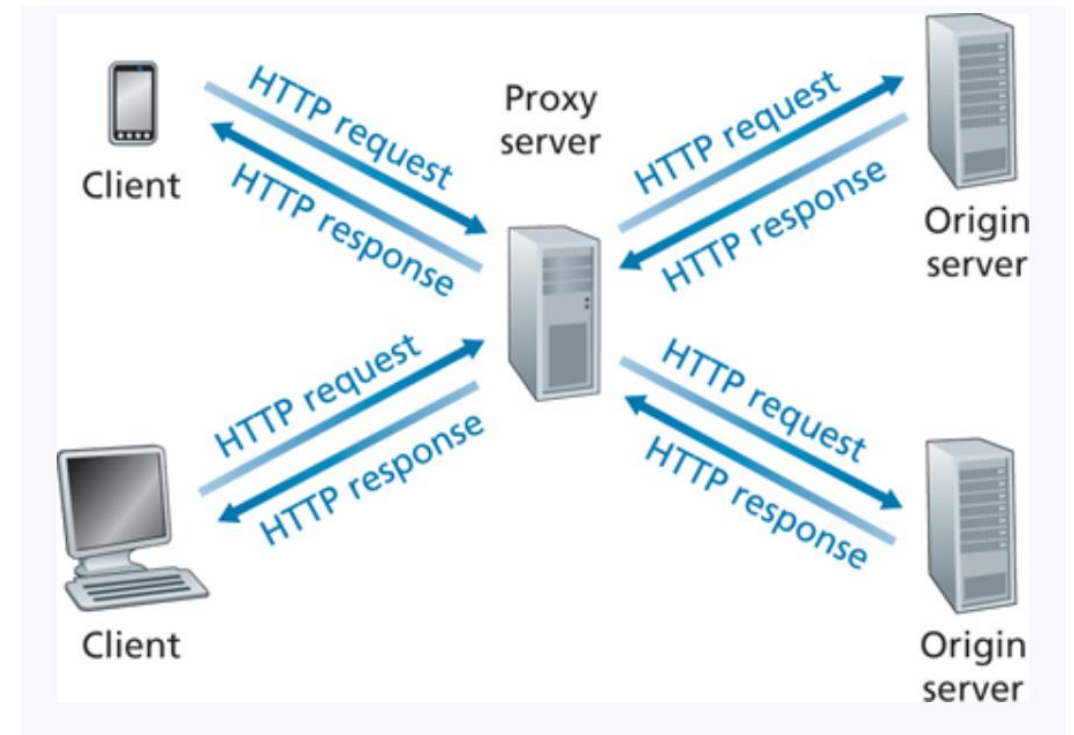


There is lots of controversy regarding the privacy and morality of tracking user information with cookies



Web Caching

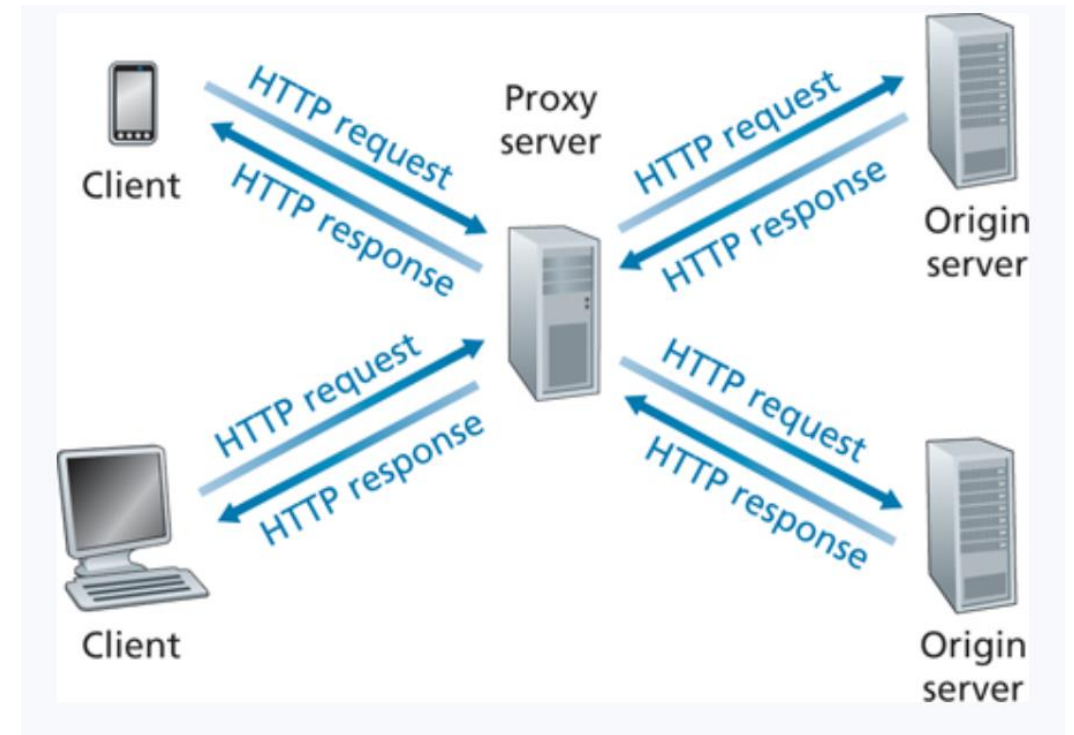
A **web cache**— also called a **proxy server**— is a network entity that satisfies HTTP requests on the behalf of an origin Web server



Web Caching

A **web cache**— also called a **proxy server**— is a network entity that satisfies HTTP requests on the behalf of an origin Web server

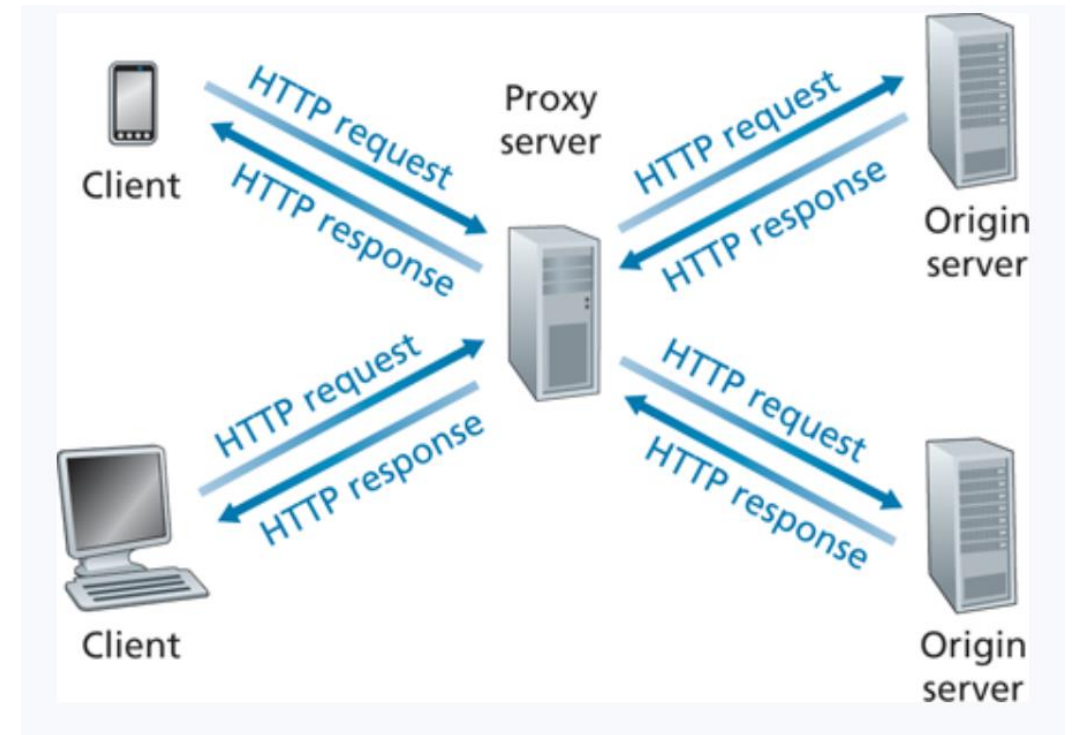
1. Browser/Client establishes a TCP connection to the Web cache and sends an HTTP request



Web Caching

A **web cache**— also called a **proxy server**— is a network entity that satisfies HTTP requests on the behalf of an origin Web server

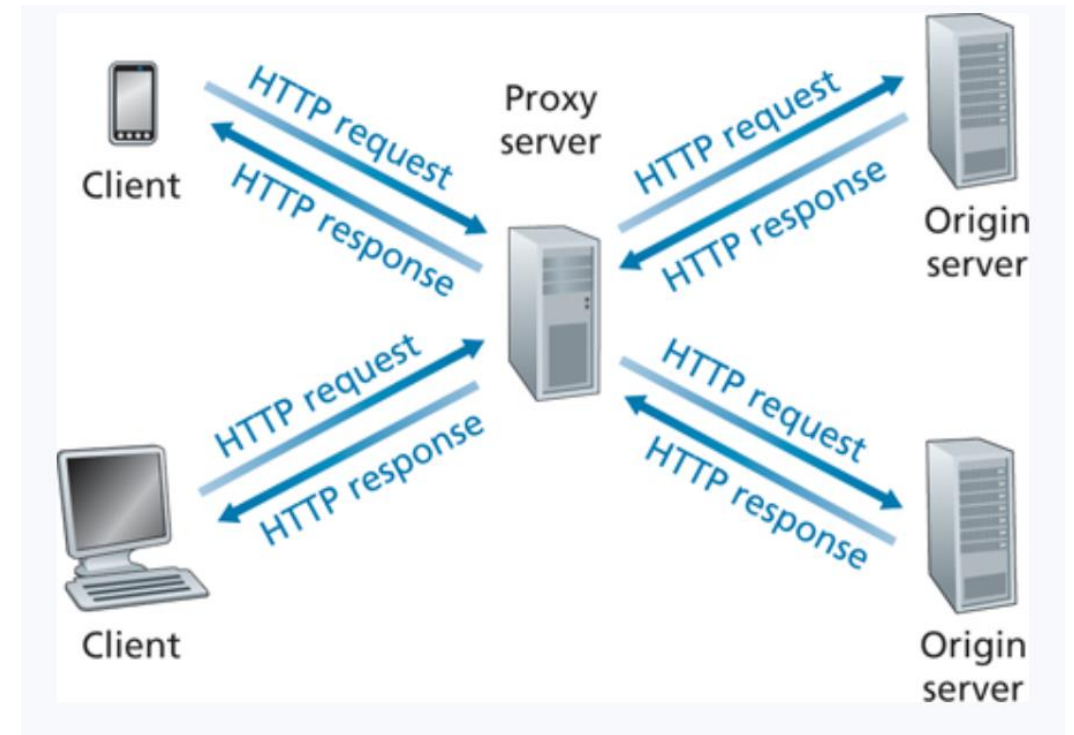
1. Browser/Client establishes a TCP connection to the Web cache and sends an HTTP request
2. Web cache checks its local storage for the requested object



Web Caching

A **web cache**— also called a **proxy server**— is a network entity that satisfies HTTP requests on the behalf of an origin Web server

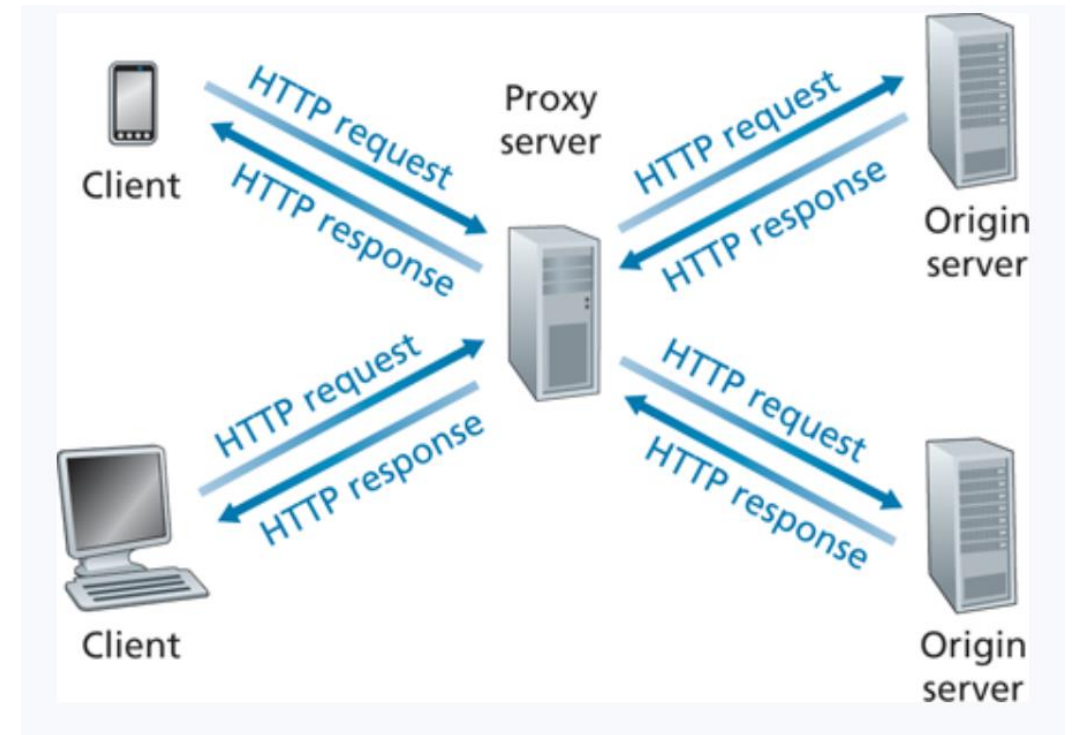
1. Browser/Client establishes a TCP connection to the Web cache and sends an HTTP request
2. Web cache checks its local storage for the requested object
3. If the web cache does not have the object, establish TCP with an origin server and issue an HTTP request



Web Caching

A **web cache**— also called a **proxy server**— is a network entity that satisfies HTTP requests on the behalf of an origin Web server

1. Browser/Client establishes a TCP connection to the Web cache and sends an HTTP request
2. Web cache checks its local storage for the requested object
3. If the web cache does not have the object, establish TCP with an origin server and issue an HTTP request
4. Web cache stores a local copy of the object, then issues an HTTP response with the object

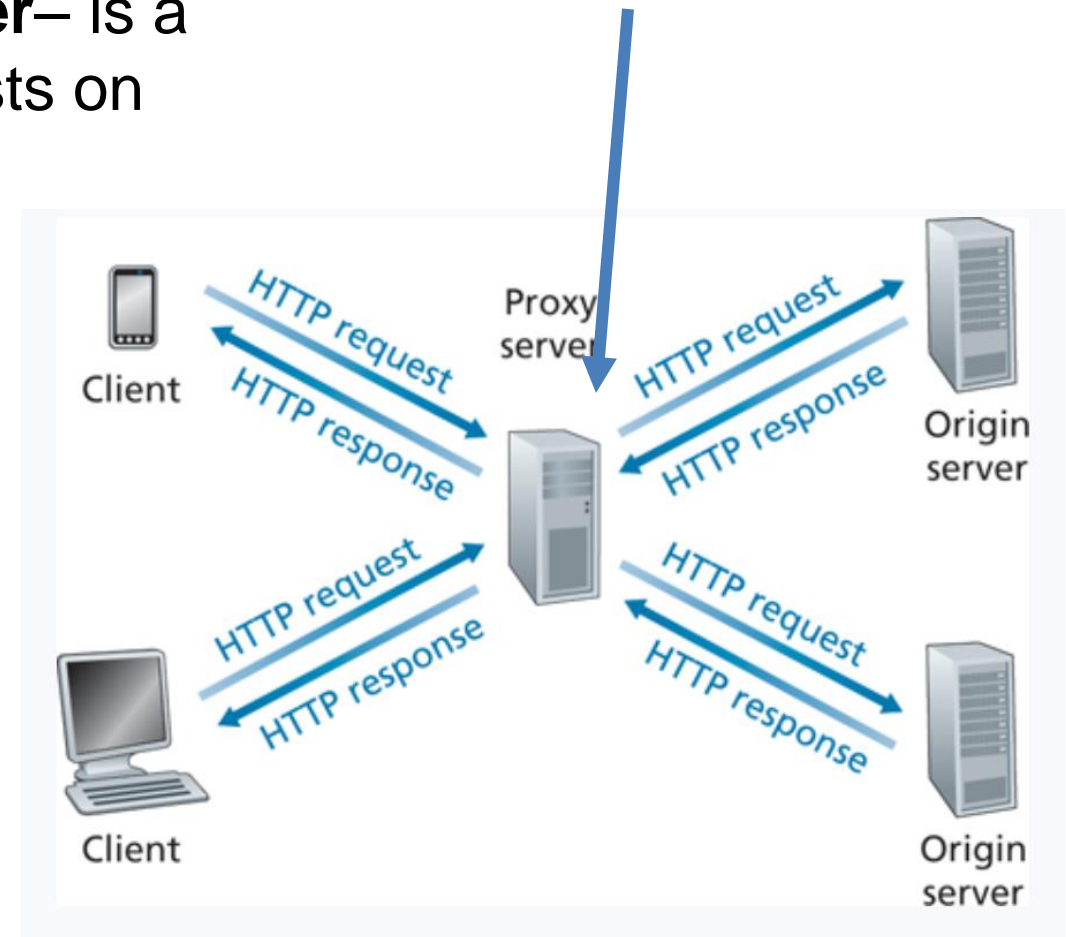


Web Caching

A **web cache**— also called a **proxy server**— is a network entity that satisfies HTTP requests on the behalf of an origin Web server

1. Browser/Client establishes a TCP connection to the Web cache and sends an HTTP request
2. Web cache checks its local storage for the requested object
3. If the web cache does not have the object, establish TCP with an origin server and issue an HTTP request
4. Web cache stores a local copy of the object, then issues an HTTP response with the object

Typically installed and maintained by an ISP



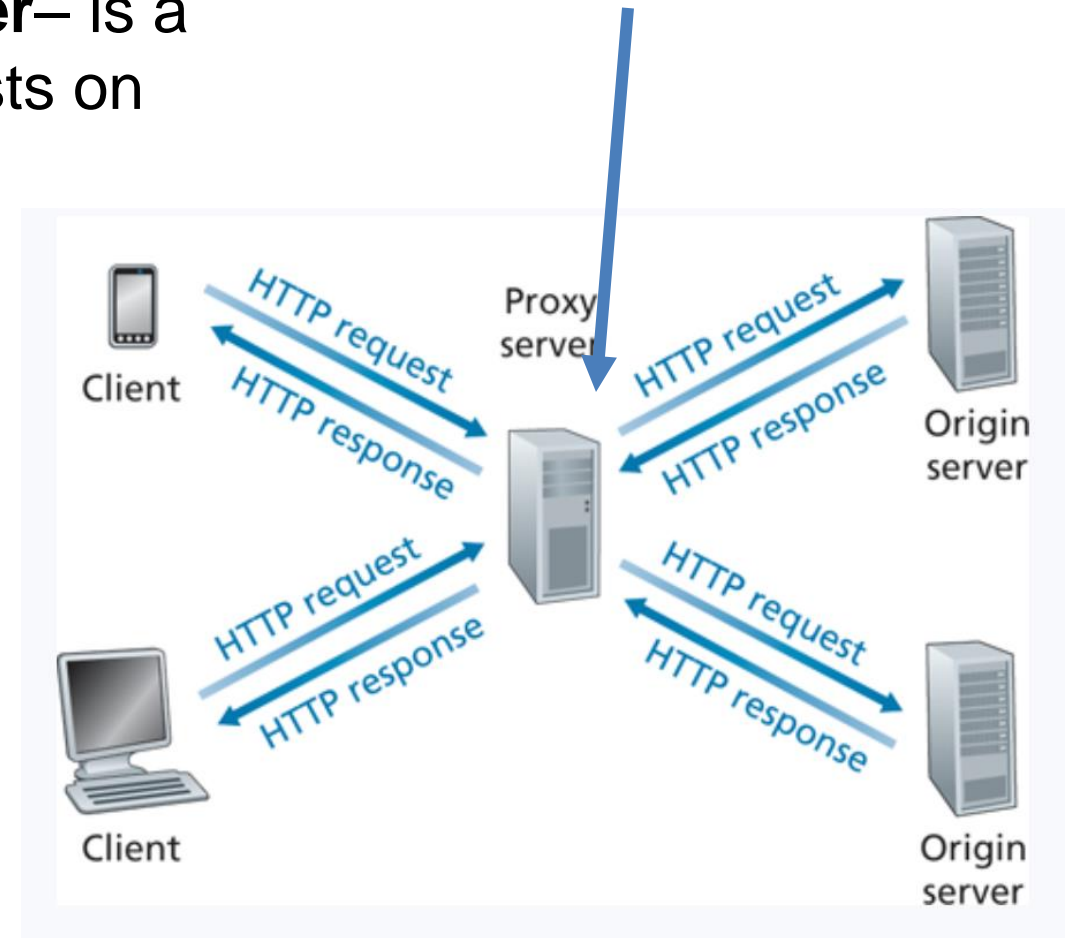
Web Caching

A **web cache**— also called a **proxy server**— is a network entity that satisfies HTTP requests on the behalf of an origin Web server

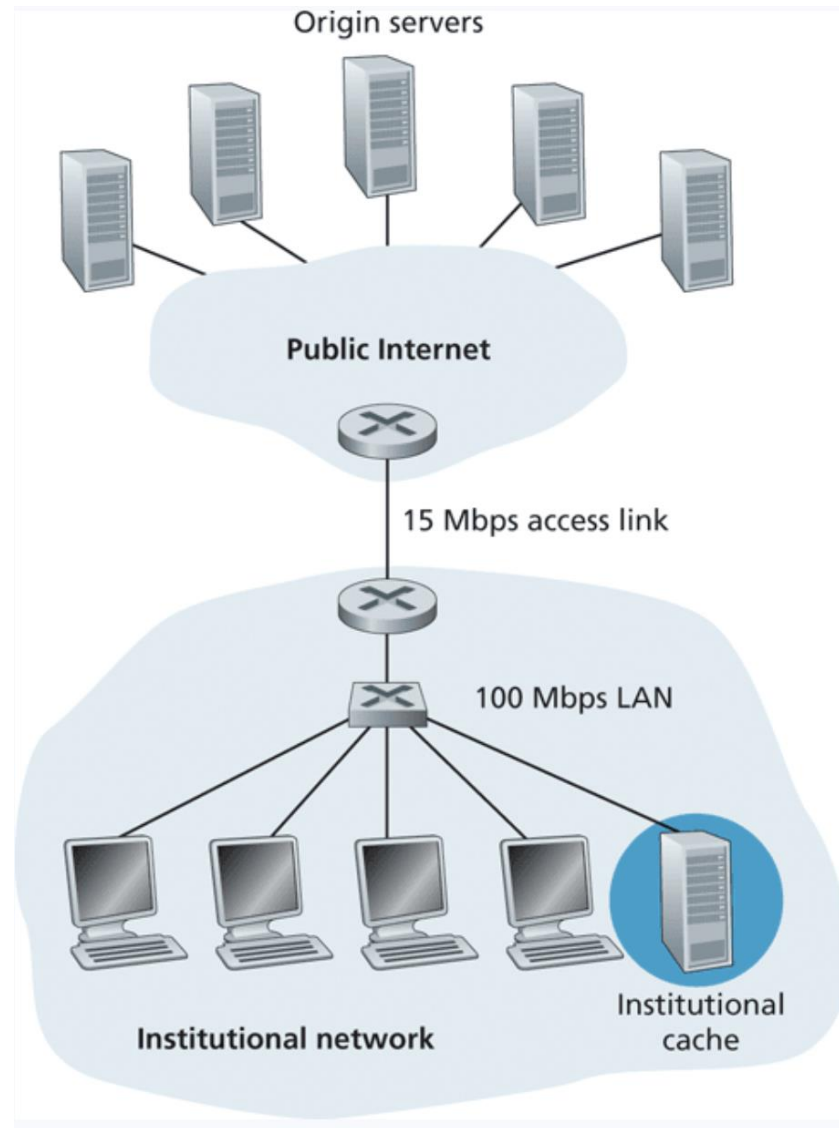
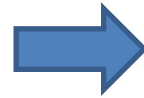
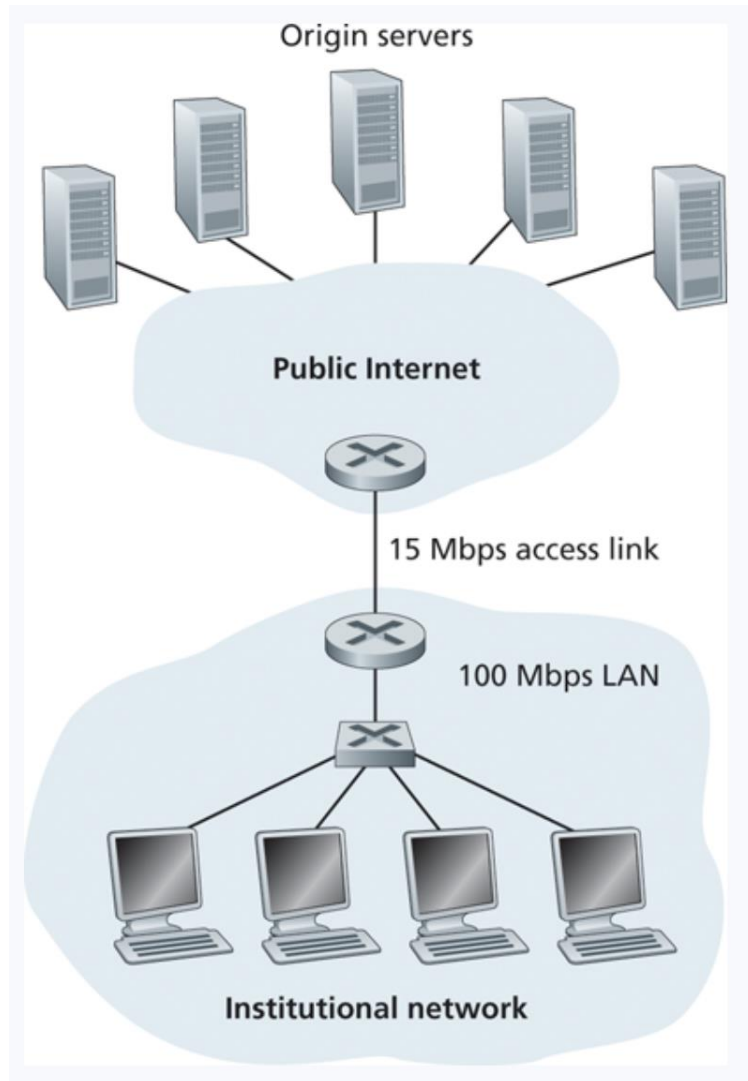
- * Improves response time (especially if the cache has the object that is requested)

- * The connection from the client to the cache is typically much faster than the connection from client to host server

Typically installed and maintained by an ISP



Web Caching



Web Caching

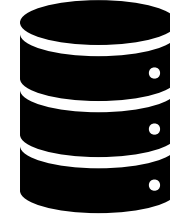


The cache might not always
have the most up to date
version in its local storage.
“Stale” objects

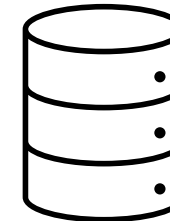
Web Caching



Proxy Server



The cache might not always have the most up to date version in its local storage.
“Stale” objects



Origin Server

Web Caching

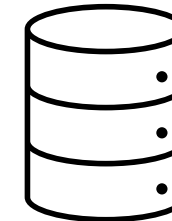


Proxy Server



The cache might not always have the most up to date version in its local storage.
“Stale” objects

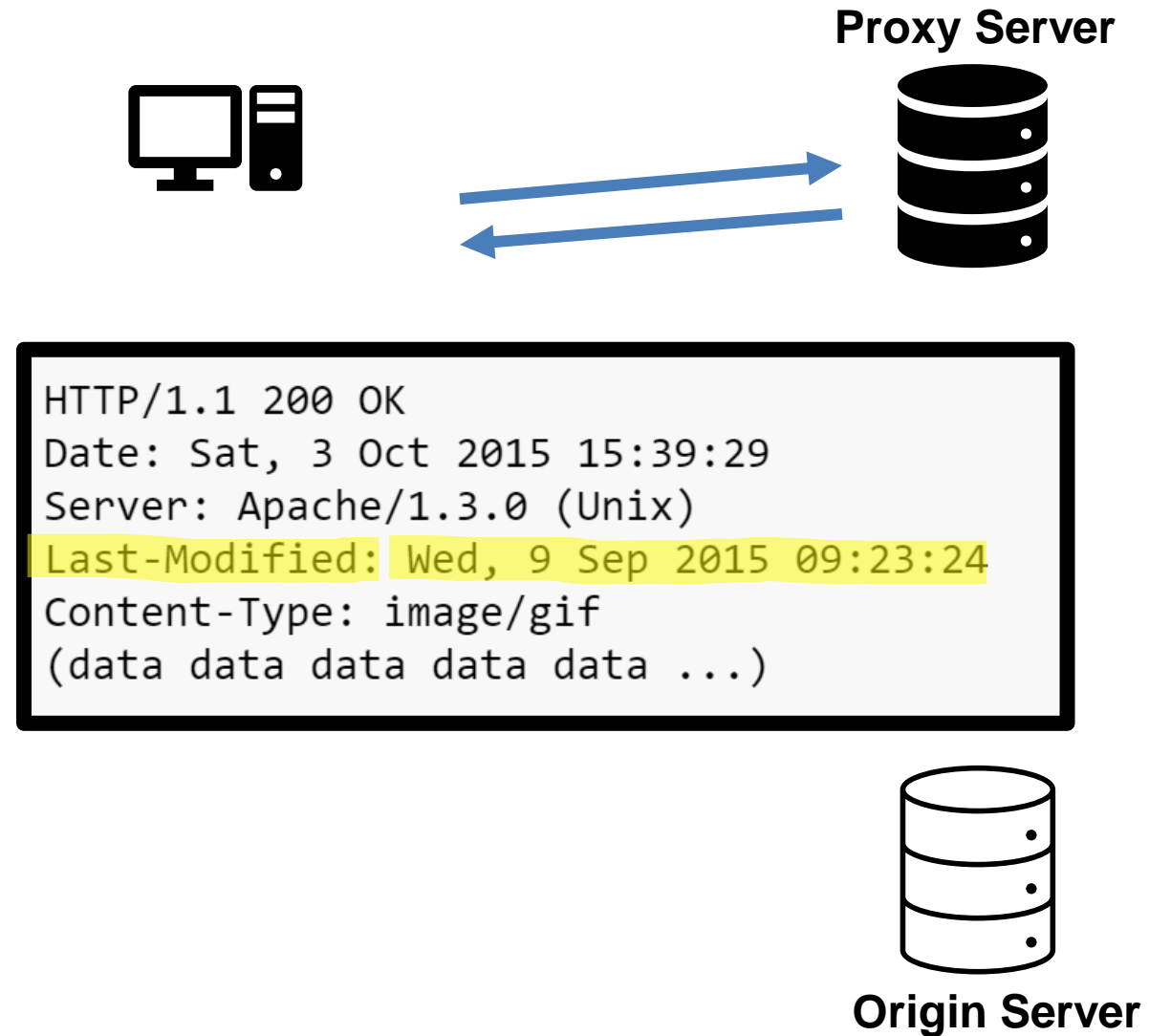
```
GET /fruit/kiwi.gif HTTP/1.1  
Host: www.exotiquecuisine.com
```



Origin Server

Web Caching

The cache might not always have the most up to date version in its local storage.
“Stale” objects



Web Caching



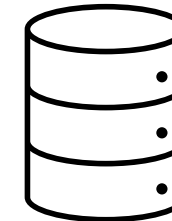
Proxy Server



The cache might not always have the most up to date version in its local storage.
“Stale” objects

```
GET /fruit/kiwi.gif HTTP/1.1
Host: www.exotiquecuisine.com
If-modified-since: Wed, 9 Sep 2015 09:23:24
```

We can issue a **conditional GET request** to retrieve an object only if it's been recently modified



Origin Server

Web Caching

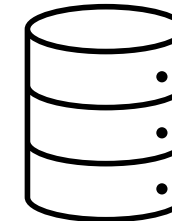
The cache might not always have the most up to date version in its local storage.
“Stale” objects

```
GET /fruit/kiwi.gif HTTP/1.1
Host: www.exotiquecuisine.com
If-modified-since: Wed, 9 Sep 2015 09:23:24
```

We can issue a **conditional GET request** to retrieve an object only if it's been recently modified



Proxy Server



Origin Server

Web Caching

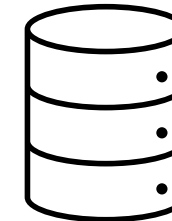
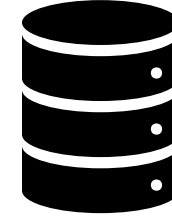
The cache might not always have the most up to date version in its local storage.
“Stale” objects

```
GET /fruit/kiwi.gif HTTP/1.1
Host: www.exotiquecuisine.com
If-modified-since: Wed, 9 Sep 2015 09:23:24
```

We can issue a **conditional GET request** to retrieve an object only if it's been recently modified



Proxy Server



Origin Server

Web Caching

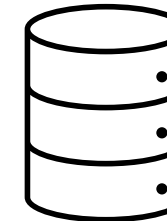
The cache might not always have the most up to date version in its local storage.
“Stale” objects

```
HTTP/1.1 304 Not Modified
Date: Sat, 10 Oct 2015 15:39:29
Server: Apache/1.3.0 (Unix)
(empty entity body)
```

We can issue a **conditional GET request** to retrieve an object only if it's been recently modified



Proxy Server



Origin Server

HTTP in action

when you use chrome developer tools to delete ads on a webpage

