CSCI 466: Networks

Lecture 4: Application Layer

Reese Pearsall

Fall 2022

*All images are stolen from the internet

OSI Model

Application Layer

Presentation Layer *

Session Layer *

Transport Layer

Network Layer

Data Link Layer

Physical Layer

Application Layer

Messages from Network Applications

↓

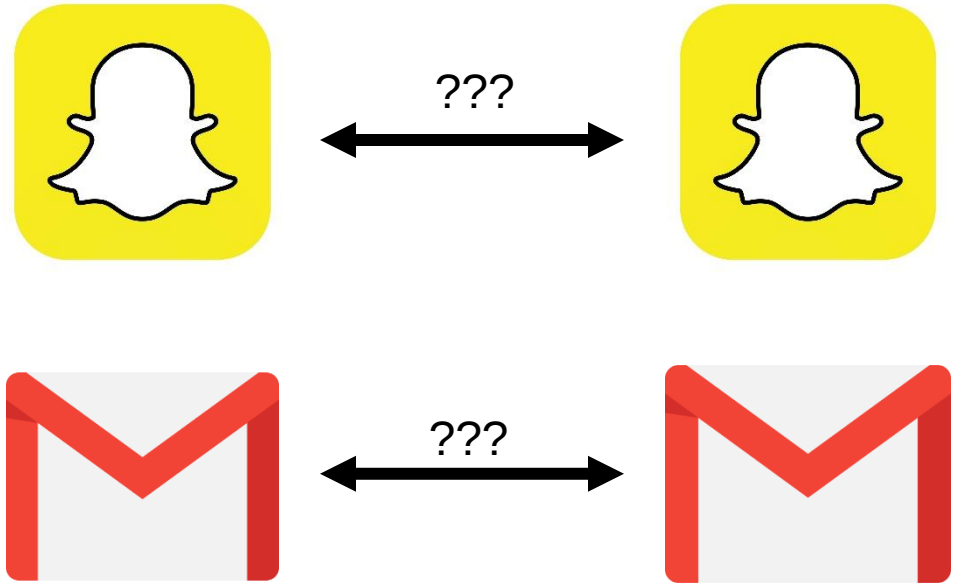
Physical Layer

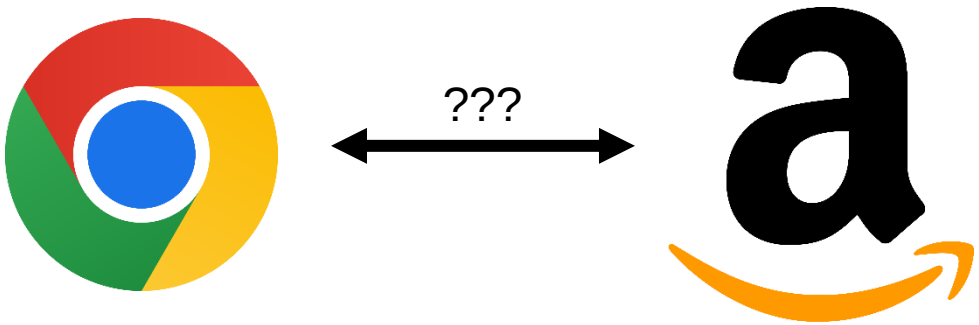
Bits being transmitted over a copper wire

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

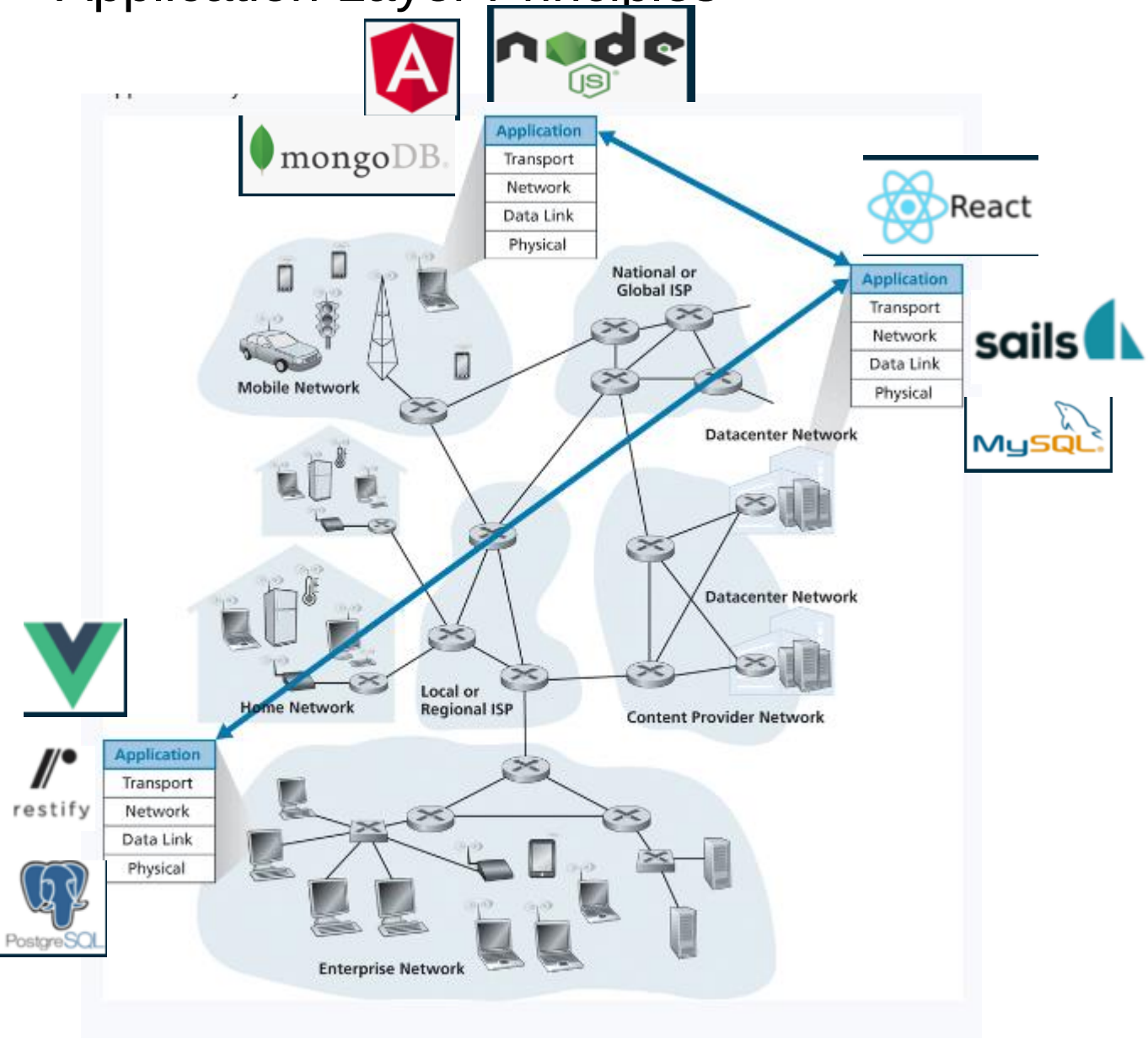
Application Layer

OSI Model





Application Layer Principles

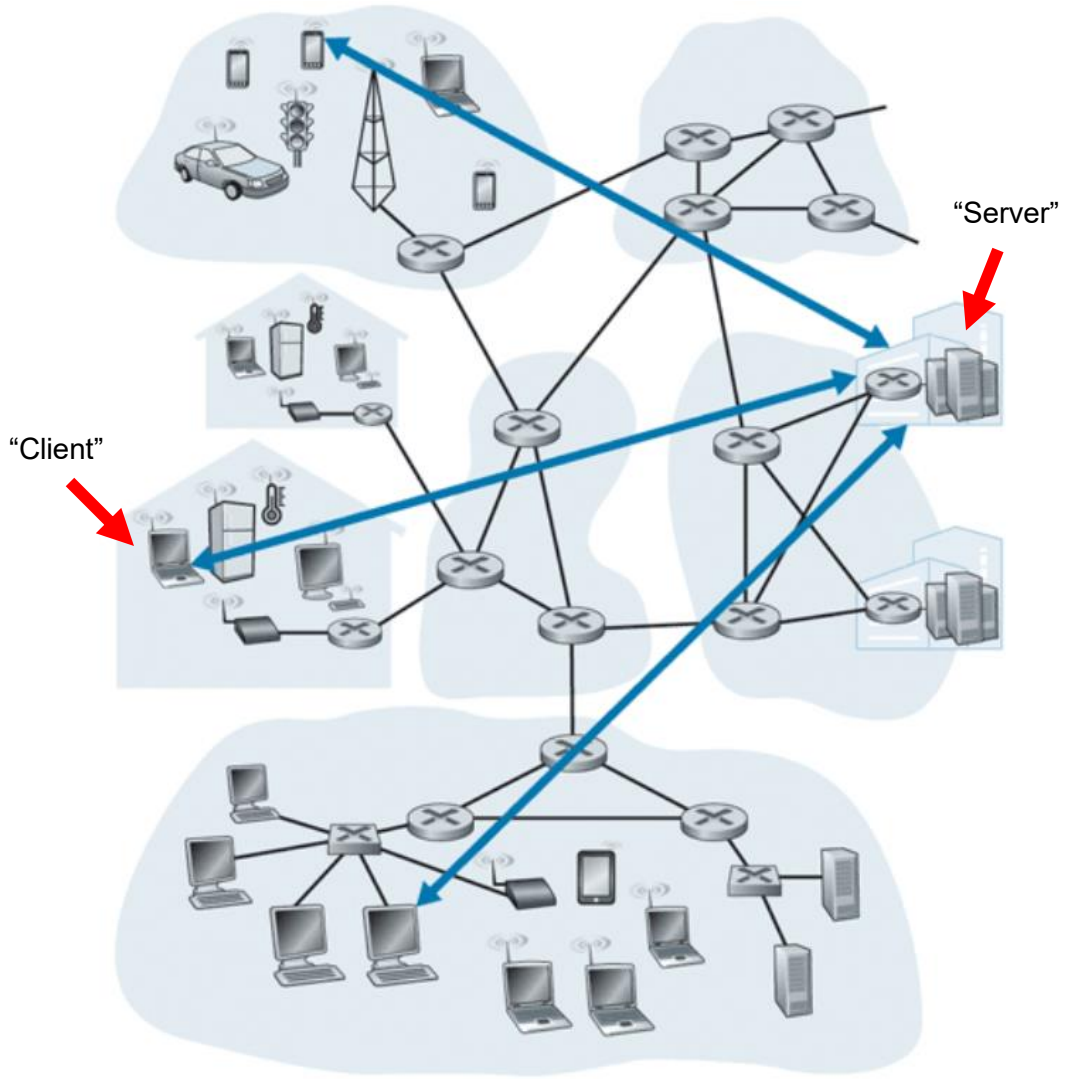


Web applications are built with different technologies

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

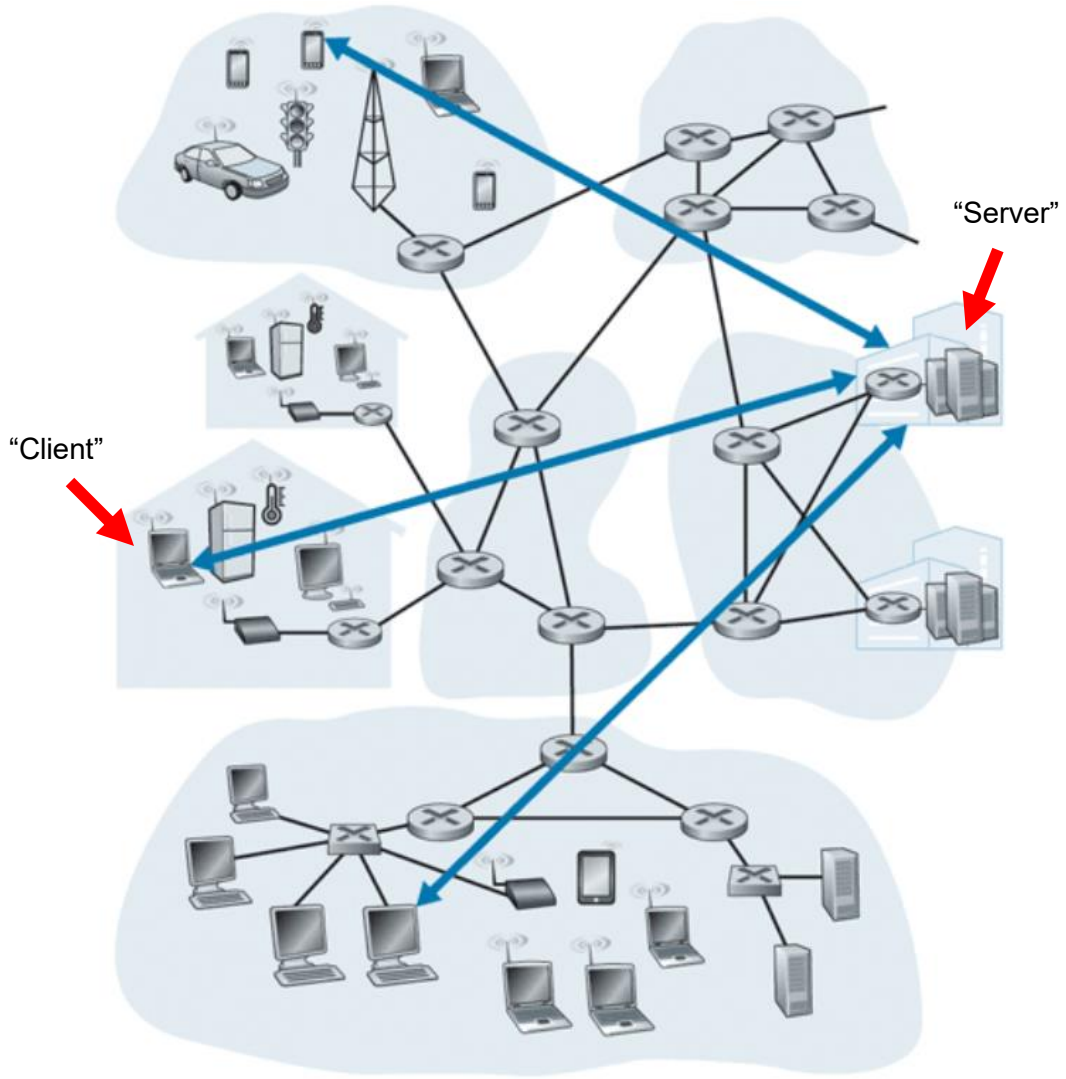
Web Application Architectures



Client-server architecture

Clients do not directly interact with each other

Web Application Architectures

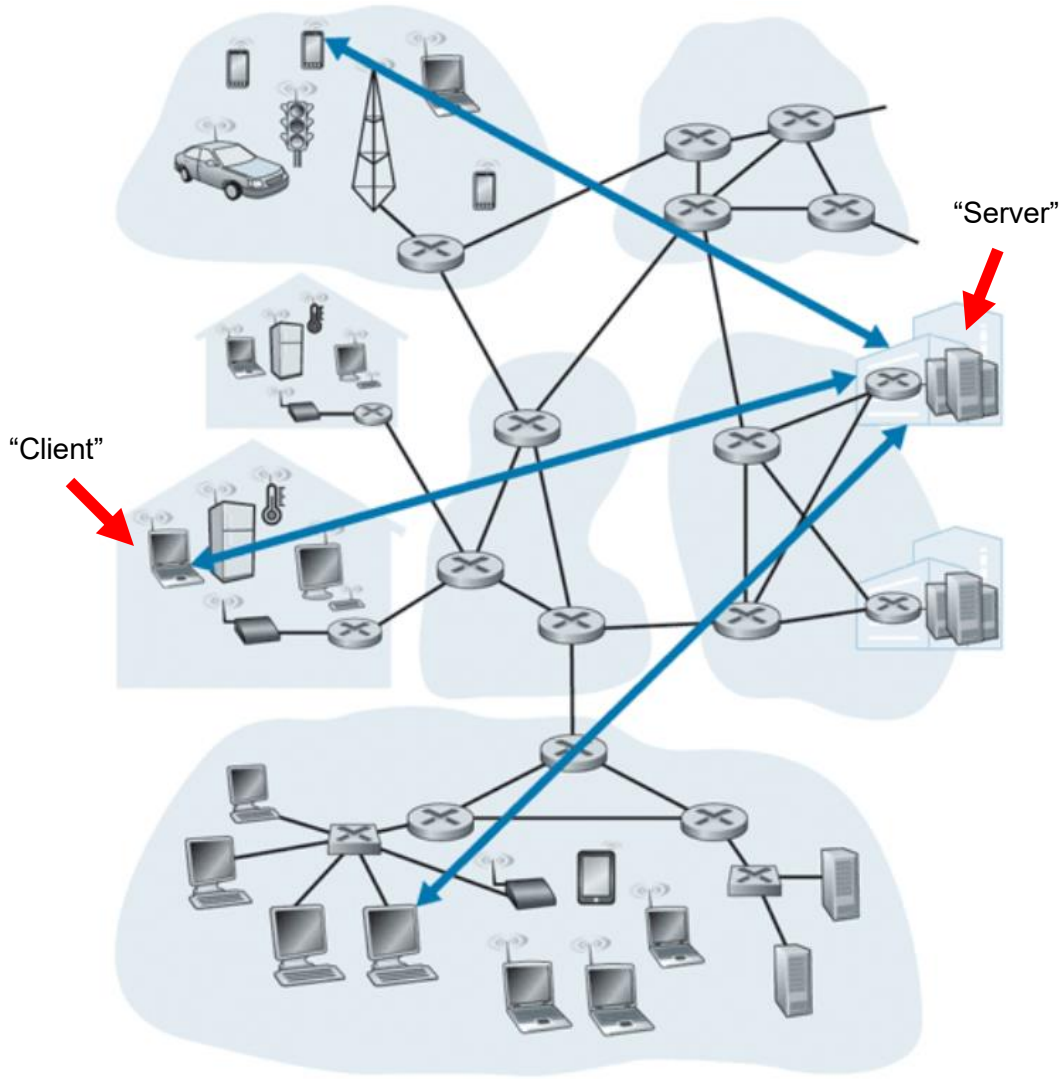


Client-server architecture

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

“Client”

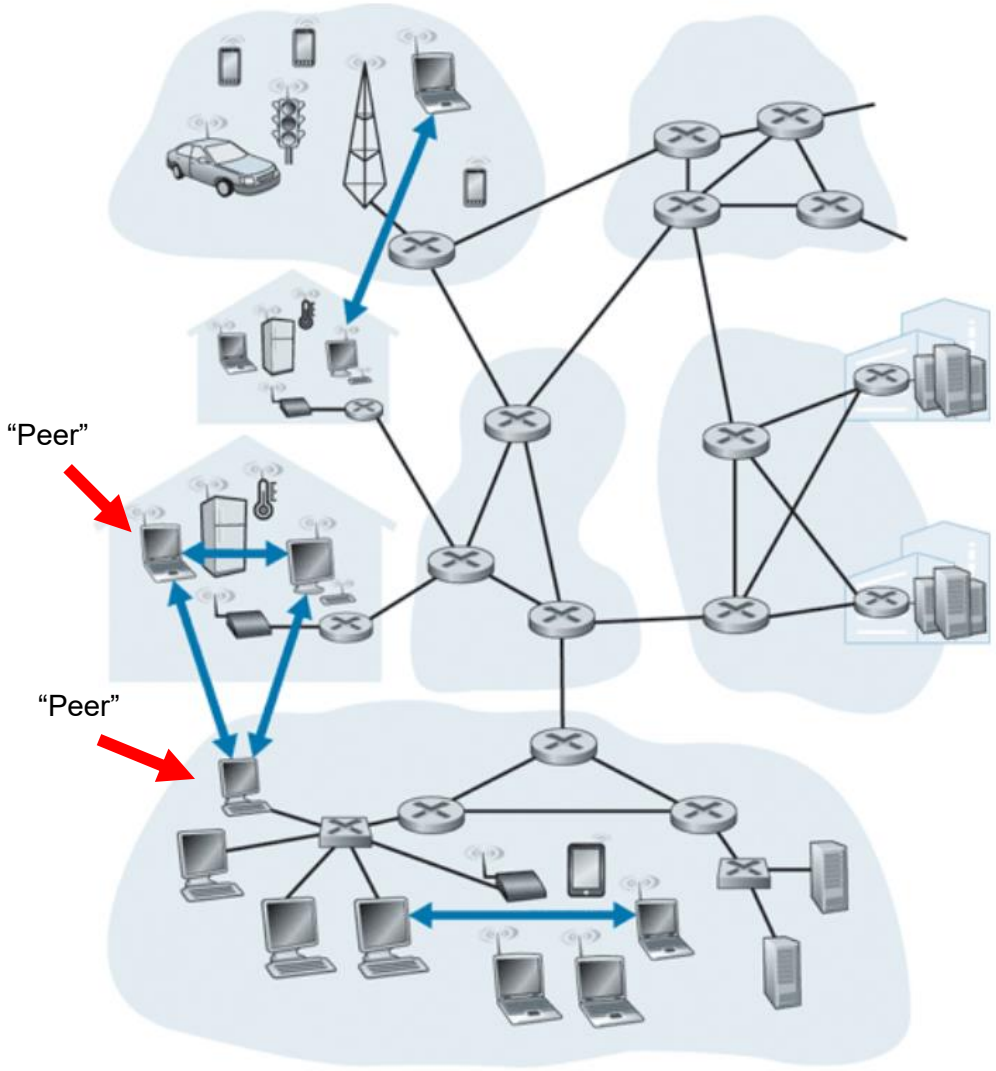
re

ch other

web browser

erver

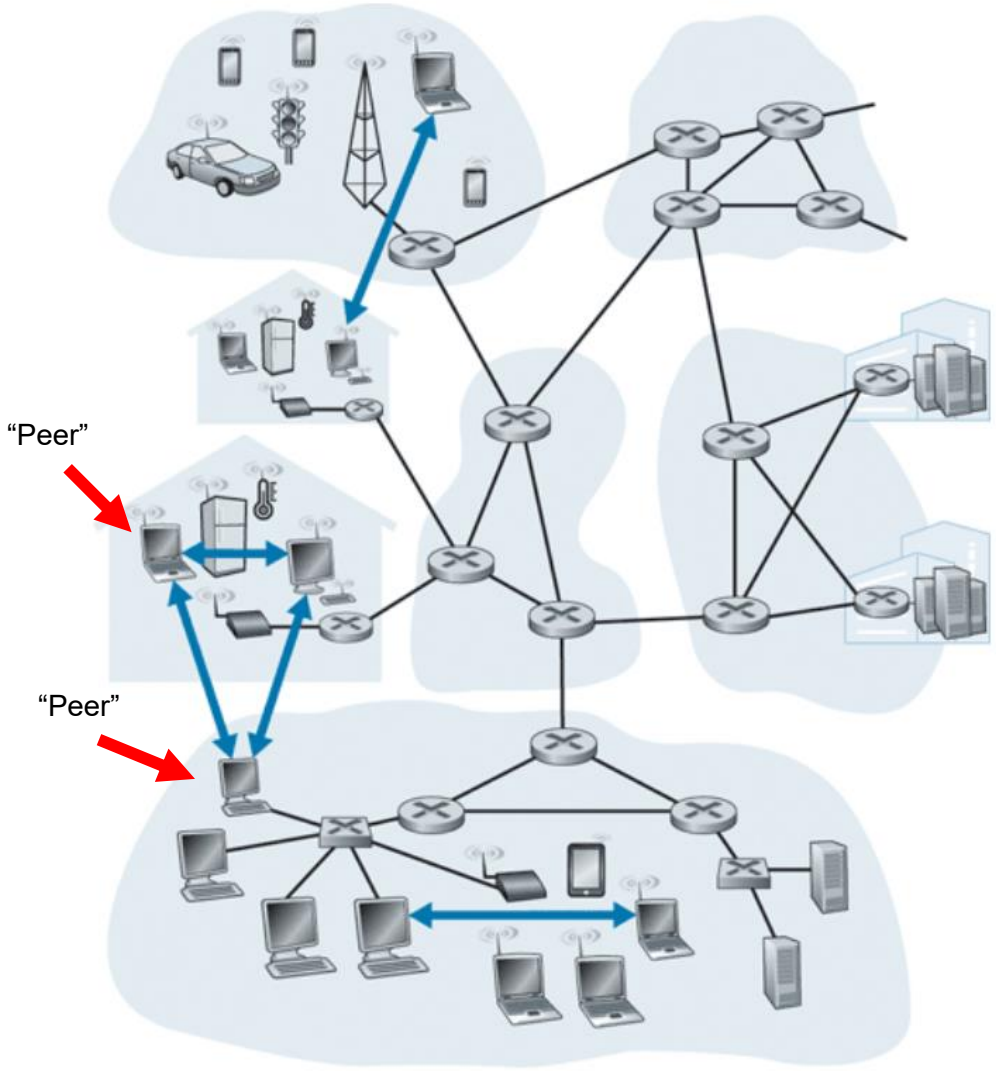
Web Application Architectures



P2P Architecture

No reliance on a dedicated server

Web Application Architectures



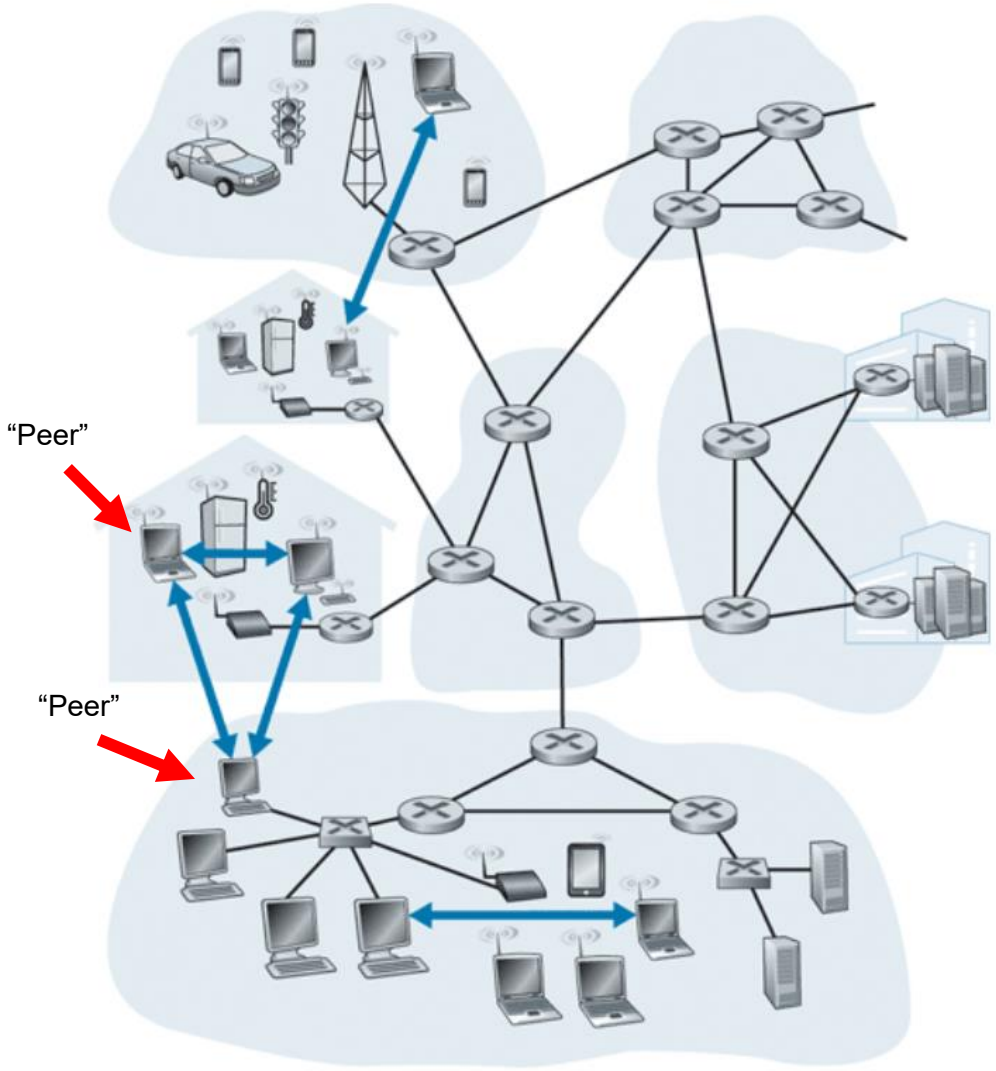
P2P Architecture

No reliance on a dedicated server

Each endpoint has same power and responsibilities.

Endpoints can be both a server and an endpoint

Web Application Architectures



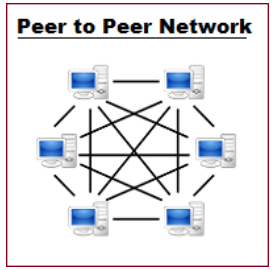
P2P Architecture

No reliance on a dedicated server

Each endpoint has same power and responsibilities.

Endpoints can be both a server and an endpoint

BitTorrent



Announcements

HW 1 is released. Due a week from today **(9/7)**

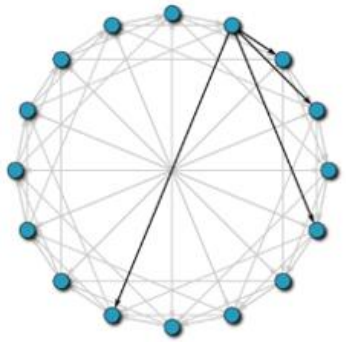
Submit it to D2L as a PDF

Web Application Architectures

Client-Server

+ Control
- Scalability
- Availability
- Distance

Peer-to-Peer



+ Cost
+ Availability
- Control
- ISP Friendliness
- Security

Application Requirements

What types of Services do different applications need from the network?

Application Requirements

What types of Services do different applications need from the network?

Application	Data Loss	Throughput	Time Sensitive

Application Requirements

What types of Services do different applications need from the network?

	no loss/loss-tolerate	*elastic/inelastic*	*yes/no*
Application	**Data Loss**	**Throughput**	**Time Sensitive**

Can the application tolerate potential loss of data?

Does the application require a specific bandwidth/throughput value?

Do endpoints need to receive data within X amount of time? Is delay allowed?

Application Requirements

What types of Services do different applications need from the network?

	no loss/loss-tolerate	*elastic/inelastic*	*yes/no*
Application	**Data Loss**	**Throughput**	**Time Sensitive**
File transfer	No loss	elastic	no

Application Requirements

What types of Services do different applications need from the network?

	no loss/loss-tolerate	*elastic/inelastic*	*yes/no*
Application	**Data Loss**	**Throughput**	**Time Sensitive**
File transfer	No loss	elastic	no
E-mail	No loss	elastic	no

Application Requirements

What types of Services do different applications need from the network?

	no loss/loss-tolerate	*elastic/inelastic*	*yes/no*
Application	**Data Loss**	**Throughput**	**Time Sensitive**
File transfer	No loss	elastic	no
E-mail	No loss	elastic	no
Web Browsing	No loss	elastic	No → yes

Application Requirements

What types of Services do different applications need from the network?

	no loss/loss-tolerate	*elastic/inelastic*	*yes/no*
Application	**Data Loss**	**Throughput**	**Time Sensitive**
File transfer	No loss	elastic	no
E-mail	No loss	elastic	no
Web Browsing	No loss	elastic	No → yes
Real-time audio	Loss-tolerant	inelastic	Yes

Application Requirements

What types of Services do different applications need from the network?

	no loss/loss-tolerate	*elastic/inelastic*	*yes/no*
Application	**Data Loss**	**Throughput**	**Time Sensitive**
File transfer	No loss	elastic	no
E-mail	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 types of Services do different applications need from the network?

	no loss/loss-tolerate	*elastic/inelastic*	*yes/no*
Application	**Data Loss**	**Throughput**	**Time Sensitive**
File transfer	No loss	elastic	no
E-mail	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 types of Services do different applications need from the network?

	no loss/loss-tolerate	*elastic/inelastic*	*yes/no*
Application	**Data Loss**	**Throughput**	**Time Sensitive**
File transfer	No loss	elastic	no
E-mail	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 types of Services do different applications need from the network?

	no loss/loss-tolerate	*elastic/inelastic*	*yes/no*	*Needed/not needed*
Application	**Data Loss**	**Throughput**	**Time Sensitive**	**Security**
File transfer	No loss	elastic	no	
E-mail	No loss	elastic	no	
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	

Transport Layer Protocols

User Datagram Prot. (UDP)

Unreliable data transfer

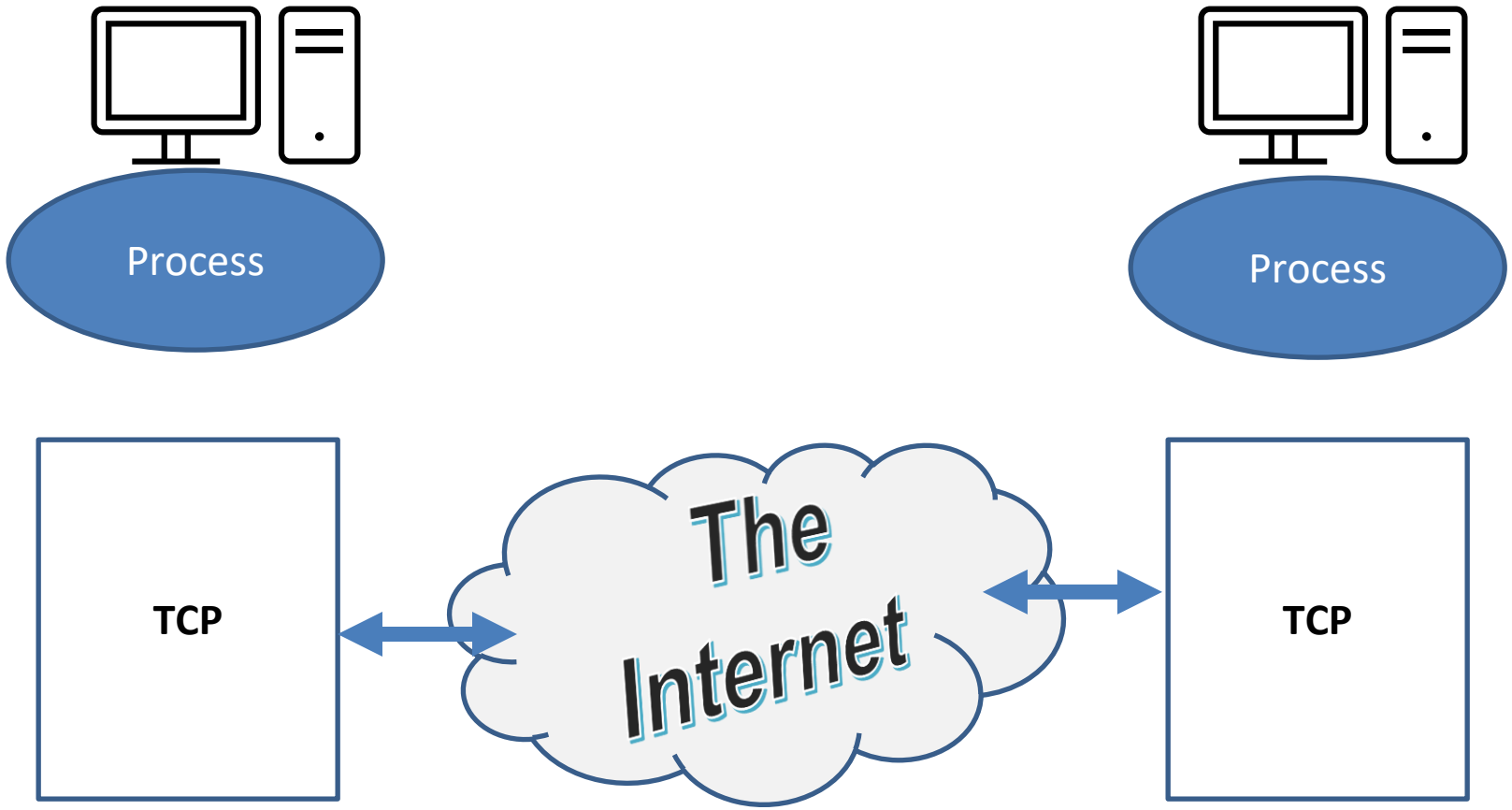
- Connection-less
  - Don’t know if receiver is present
- No flow control
  - Overflow at receiver possible
- No congestion control
  - Sender can overload the network
- No guarantees on
  - End-to-end delay
  - Throughput
  - Security

Transmission Control Prot. (TCP)

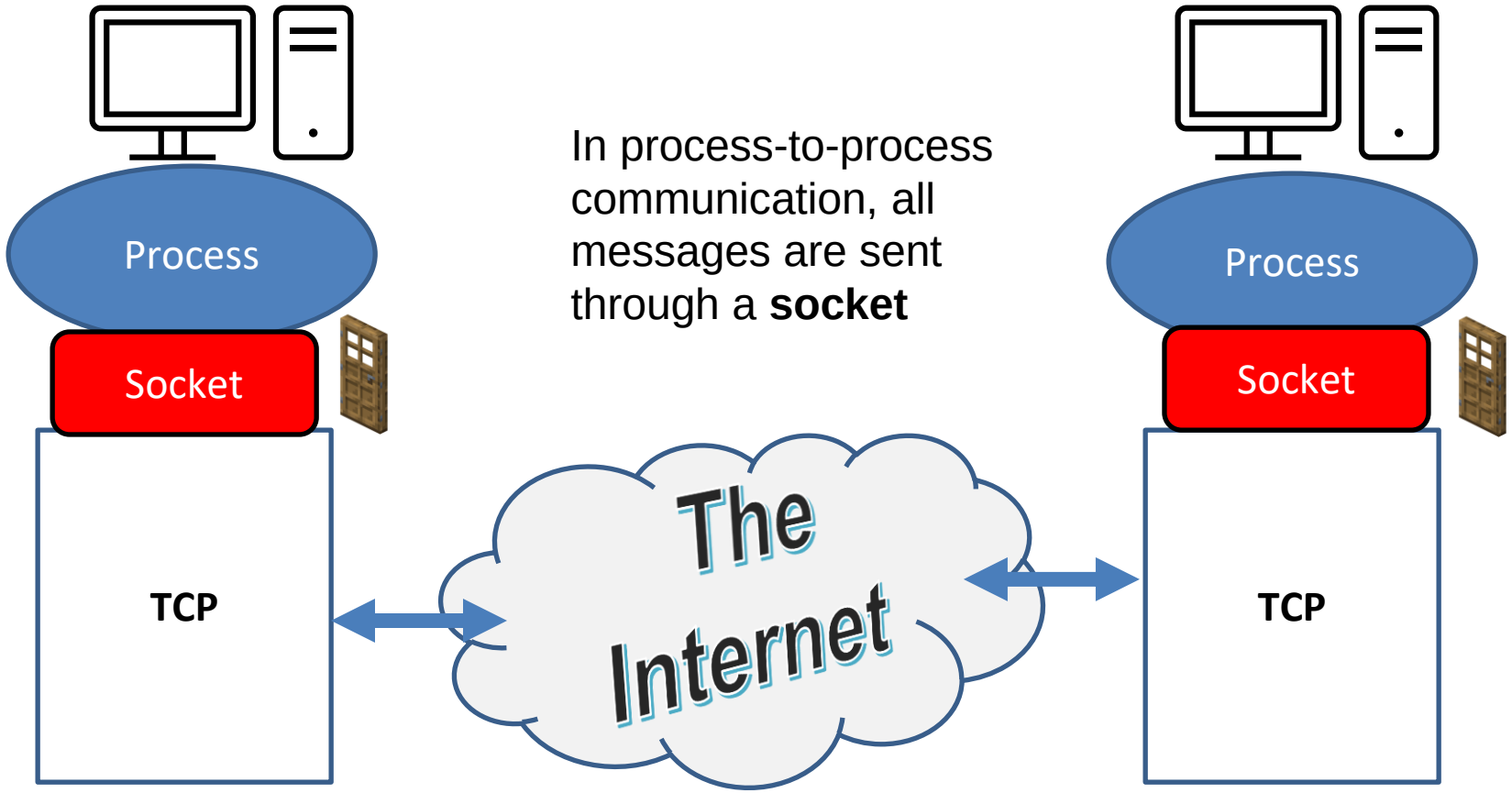
Reliable stream transport

- 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

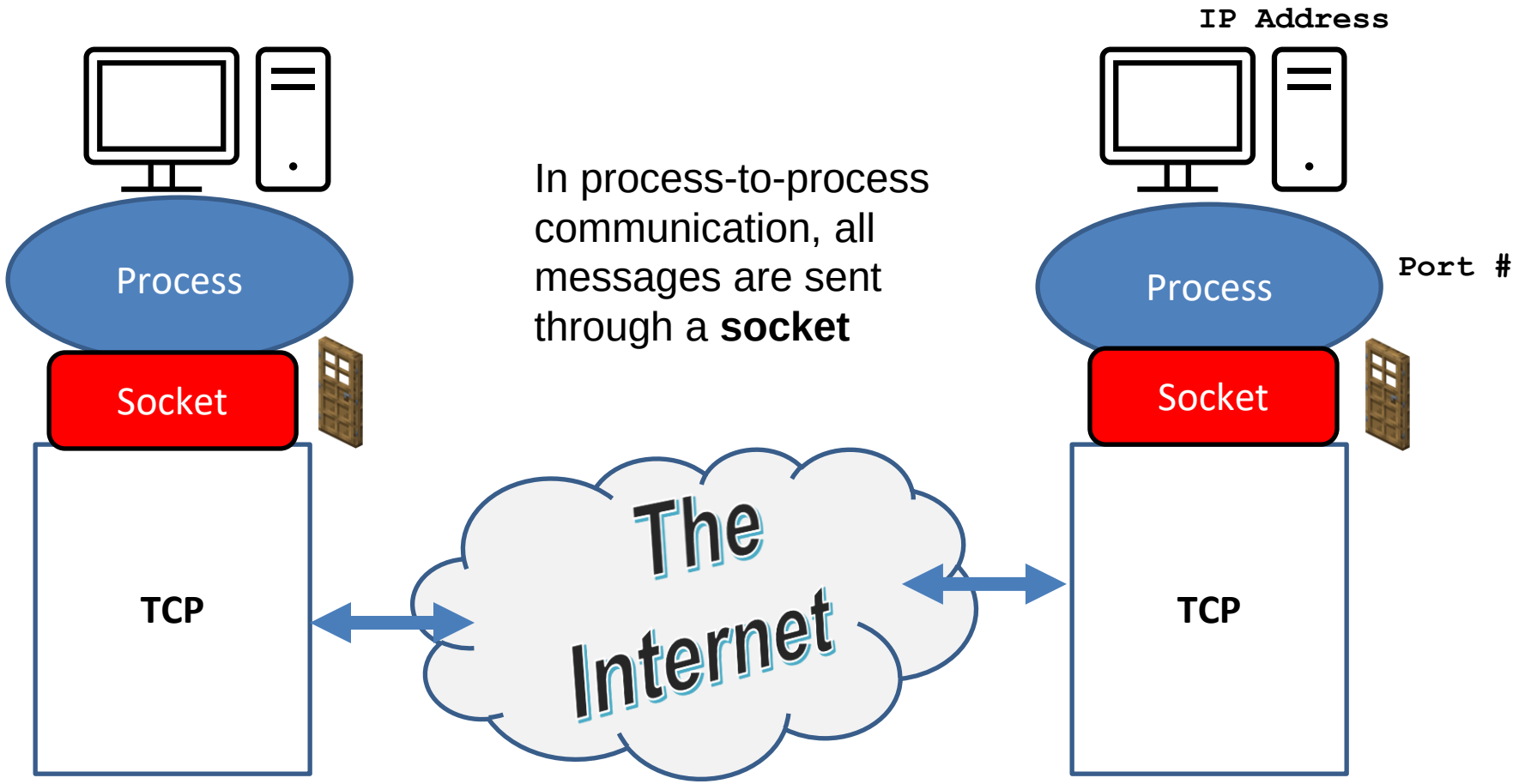


Application to Transport Interface



In process-to-process communication, all messages are sent through a **socket**

Application to Transport Interface



In process-to-process communication, all messages are sent through a **socket**

Application Layer Protocols

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

“The language of the web”

Application Layer Protocols

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
```

Application Layer Protocols

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

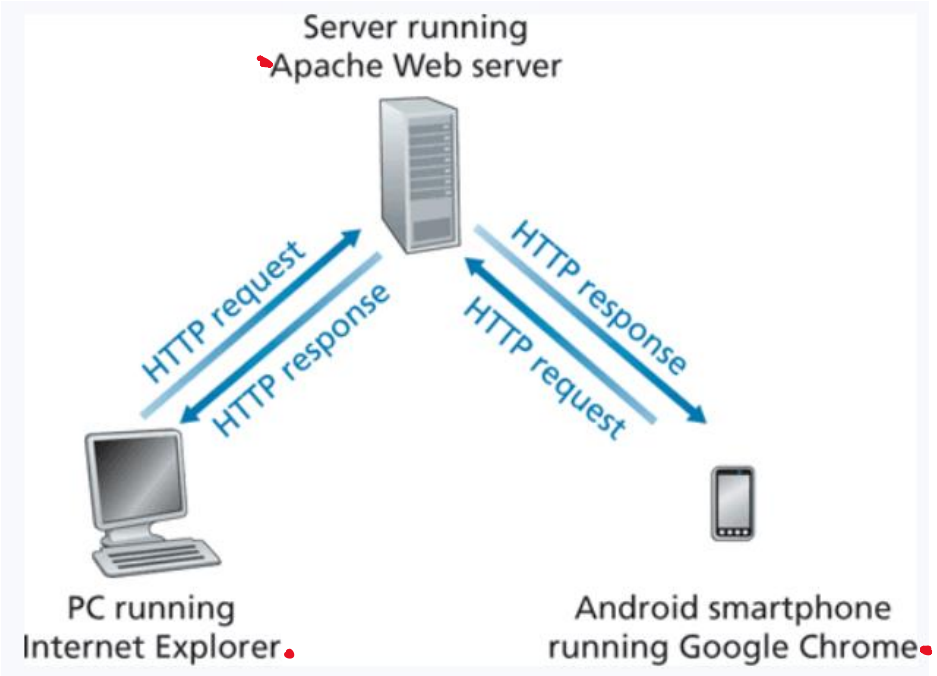
```
scheme://domain:port/path_to_object?query_string

http://cs.montana.edu/pearsall/classes/fall2022/466/main.html
```

Object

HTTP

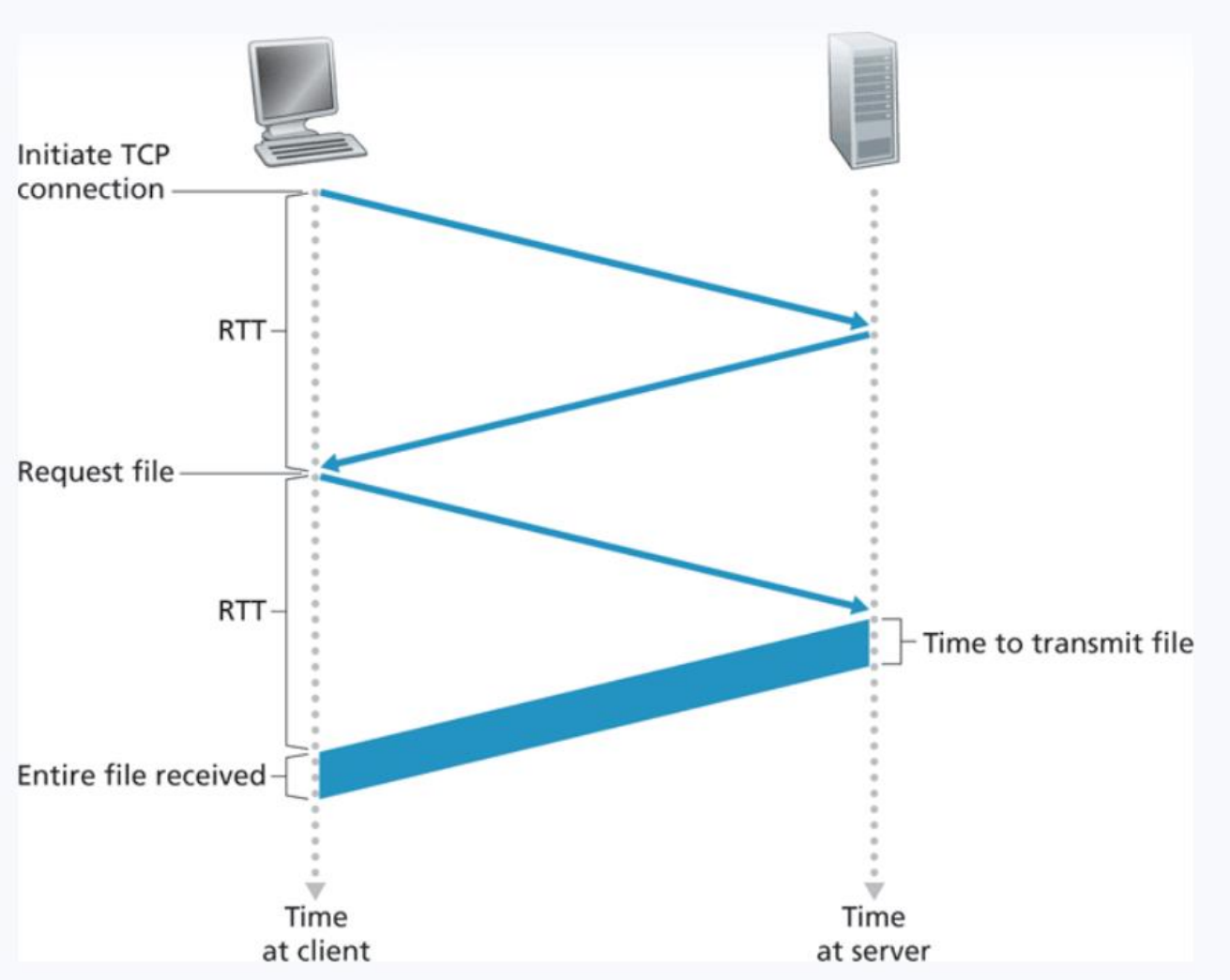
HTTP is *stateless* protocol that acts as a uniform API for different platforms



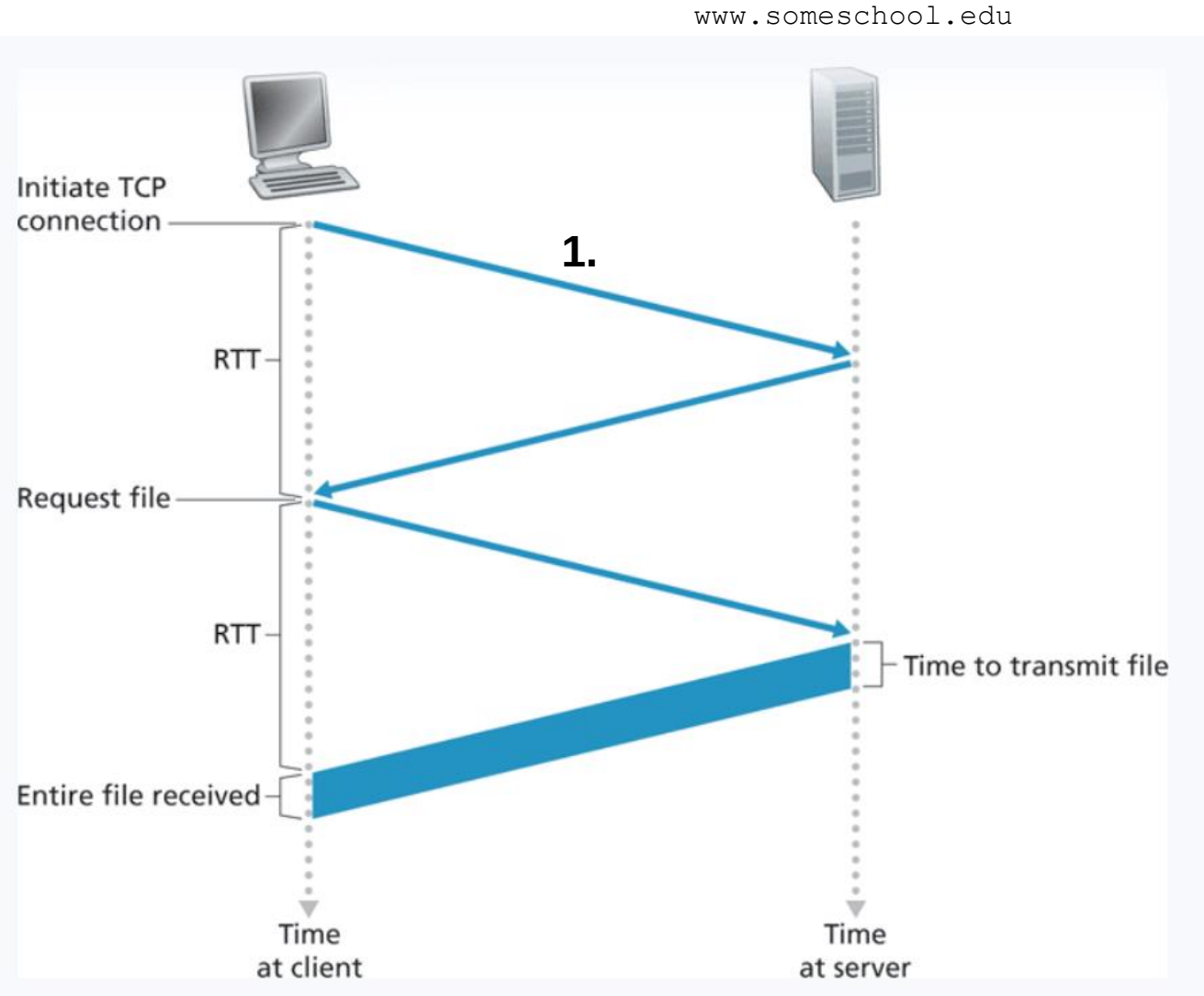
Built on TCP

HTTP

www.someschool.edu

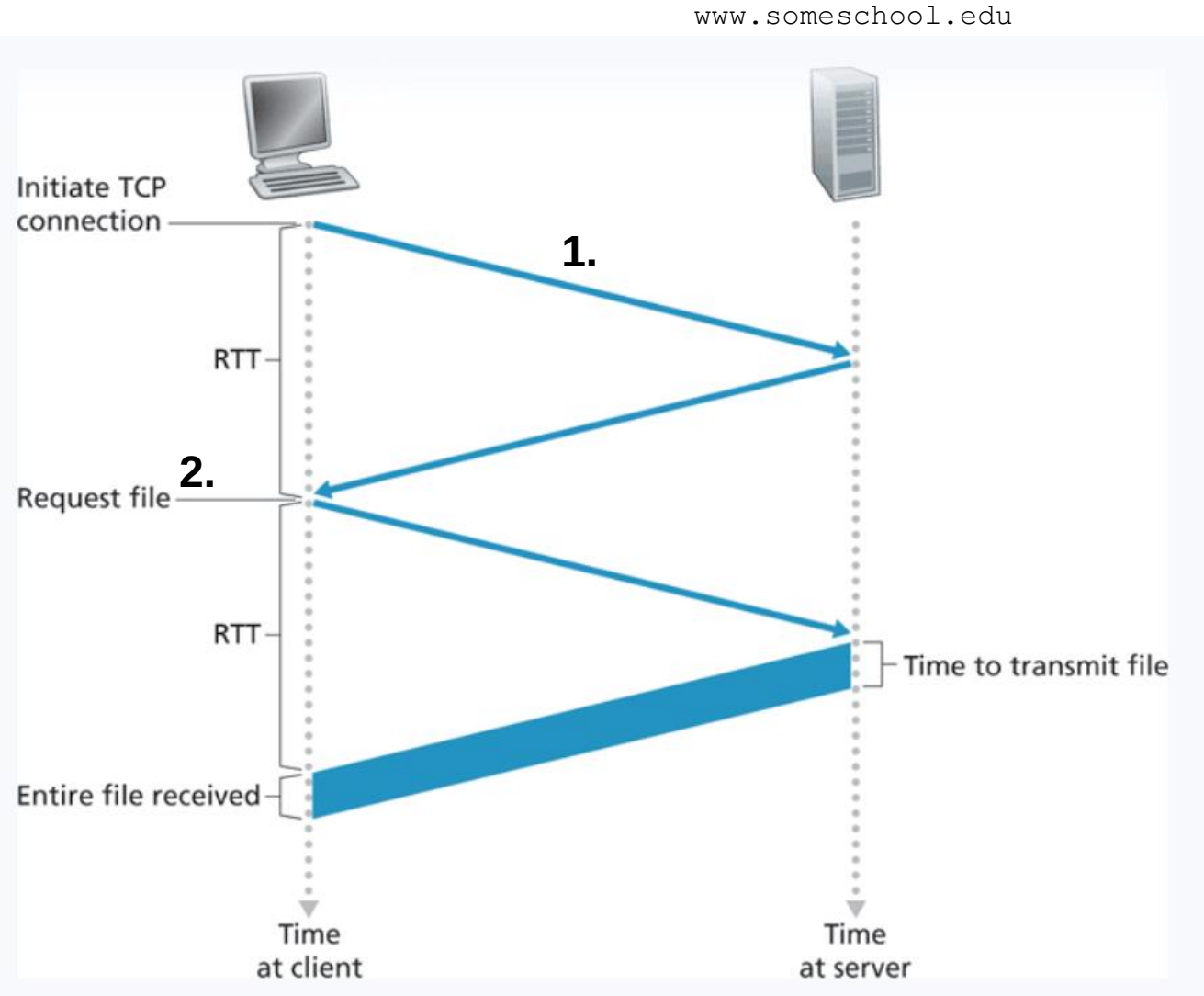


HTTP



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

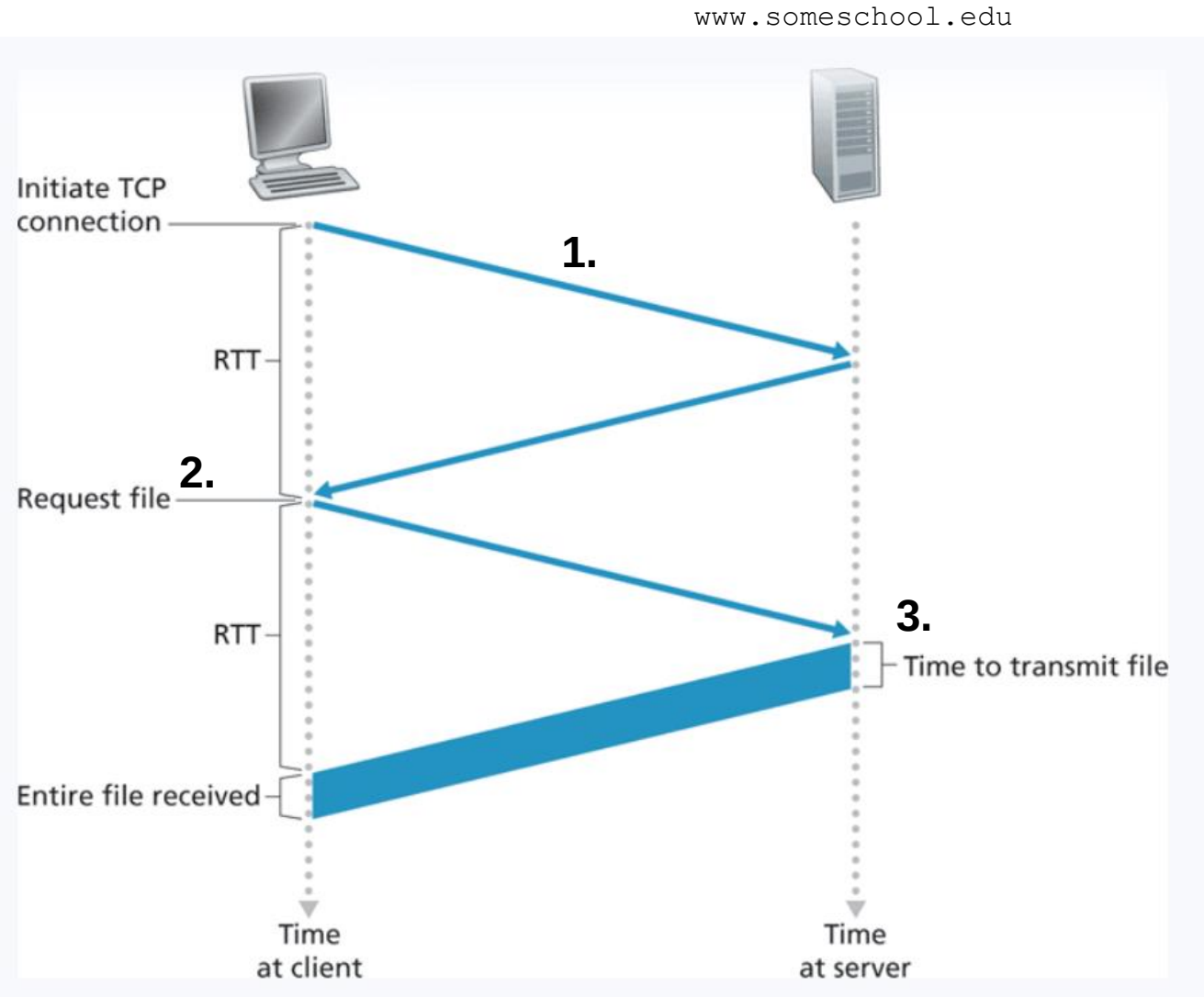
HTTP



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

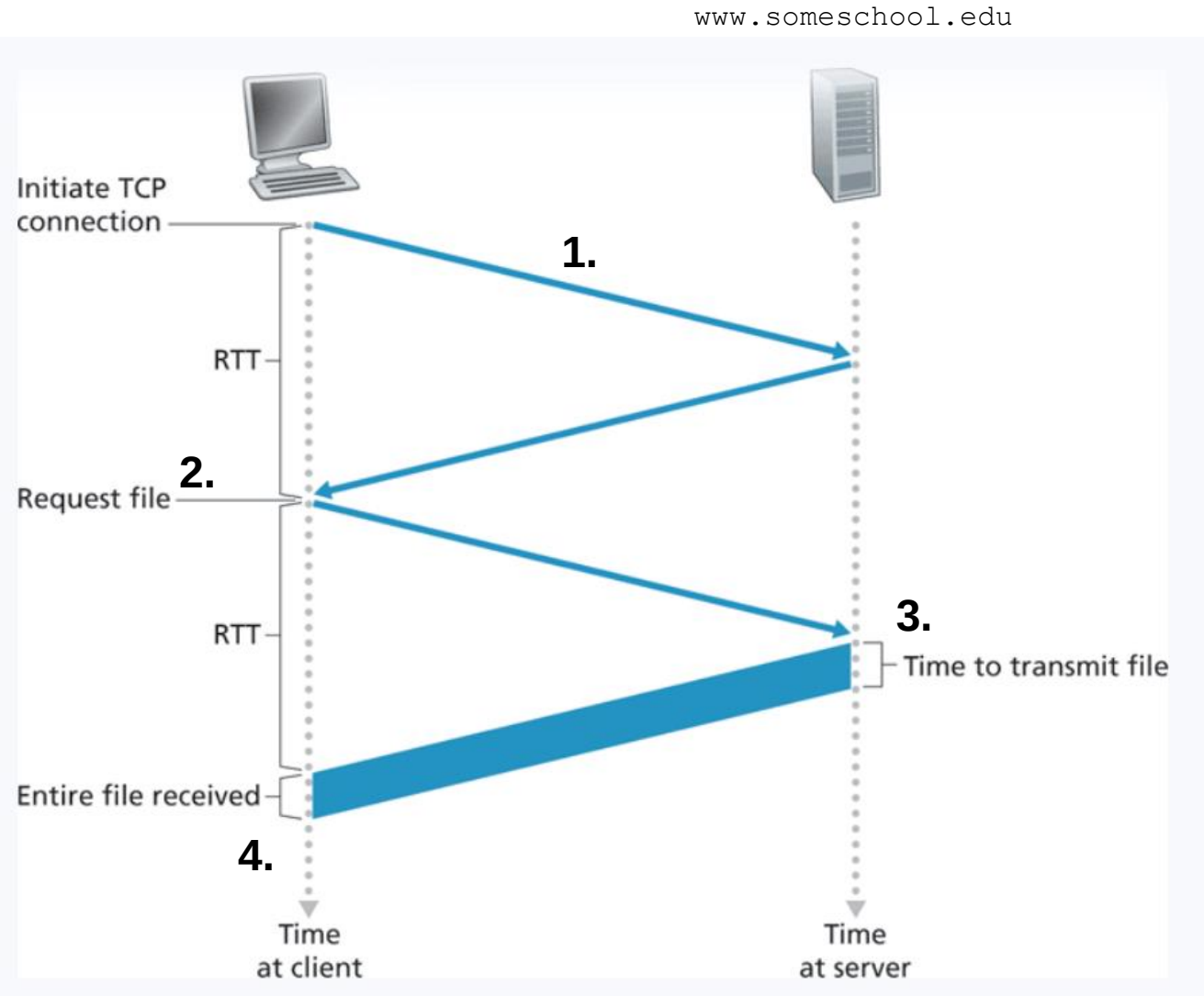


1. HTTP client process initiates a TCP connection to the server on port number 80. A socket is created at both the client and the server

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



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

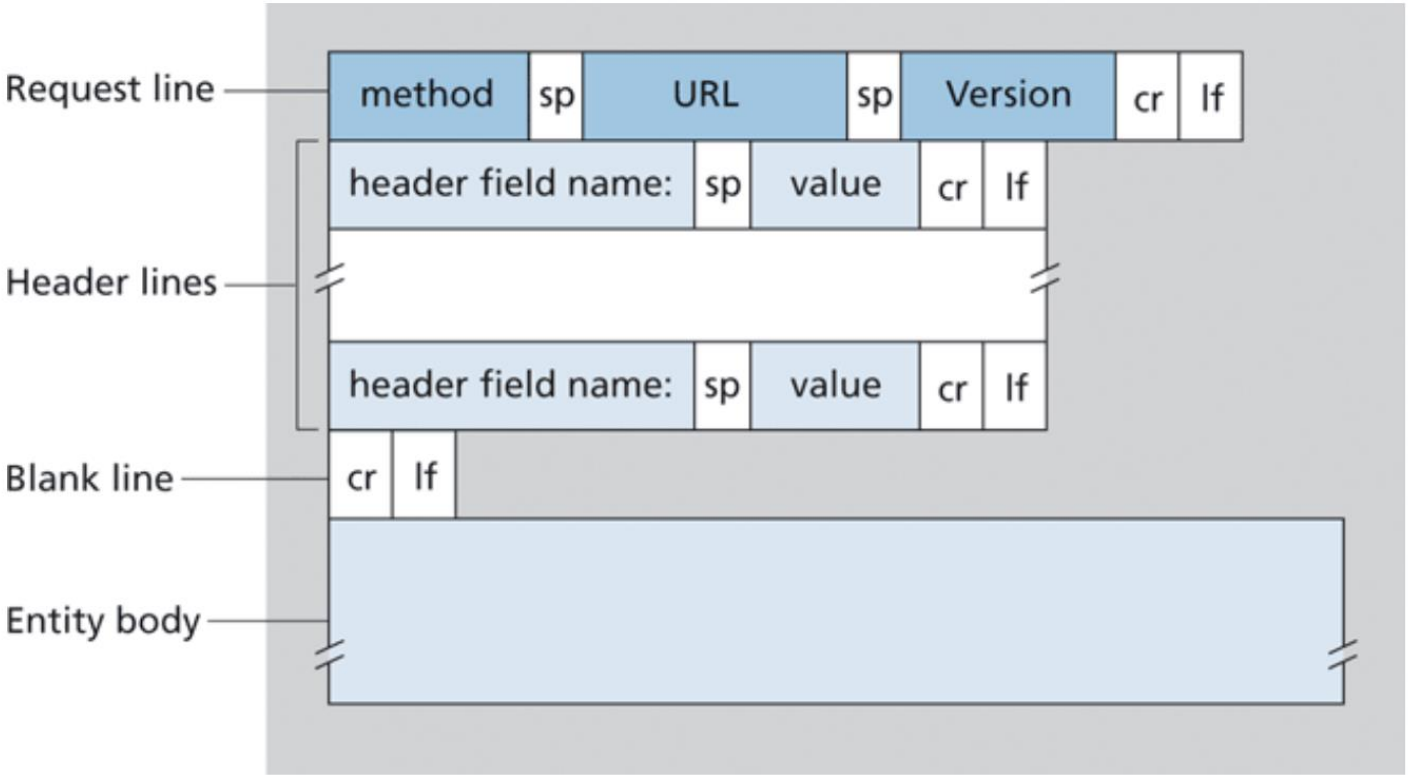
2. The HTTP client sends an HTTP request to the server via its socket. This request includes the path name for the object

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

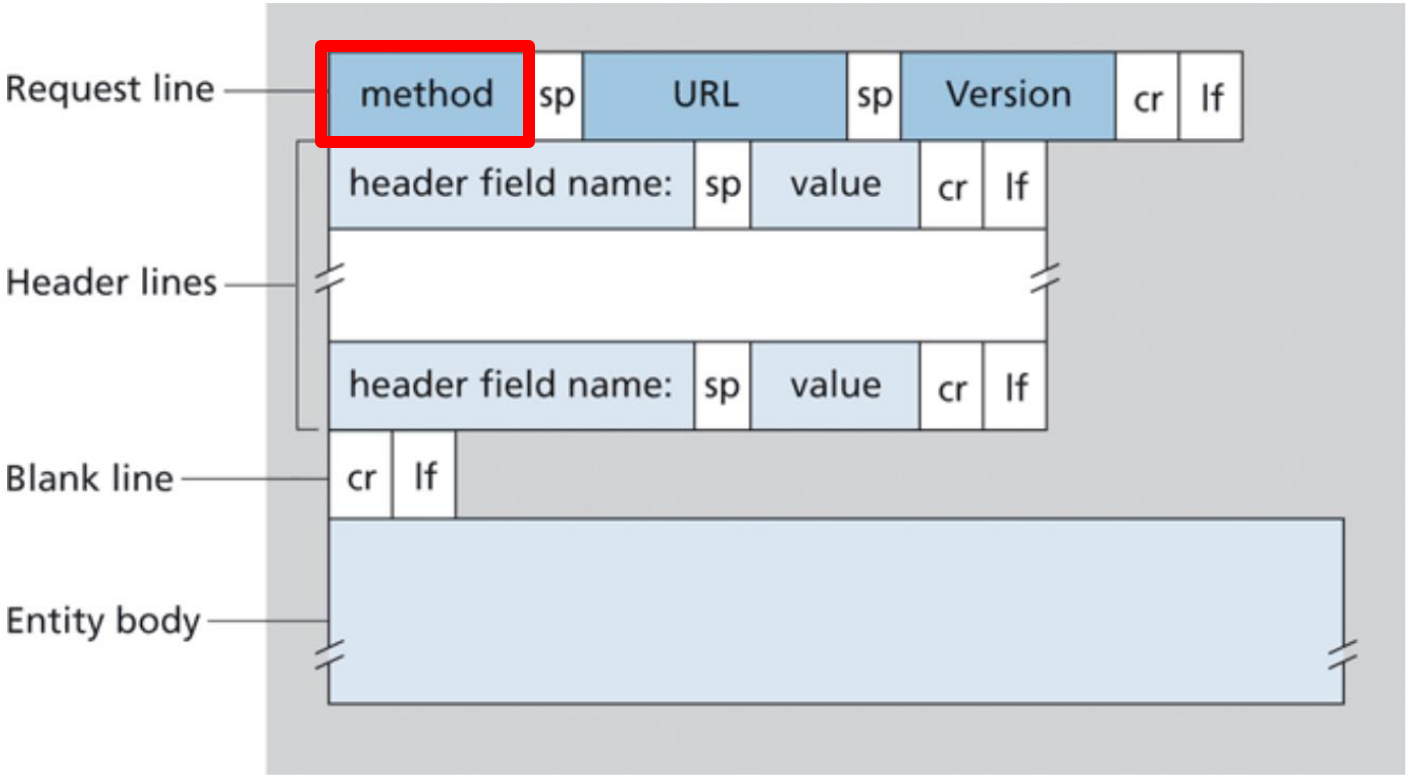
4. Client receives the HTTP response. Once entire file is received, then TCP connection is closed

(repeat for each web object)

HTTP Request Message Format

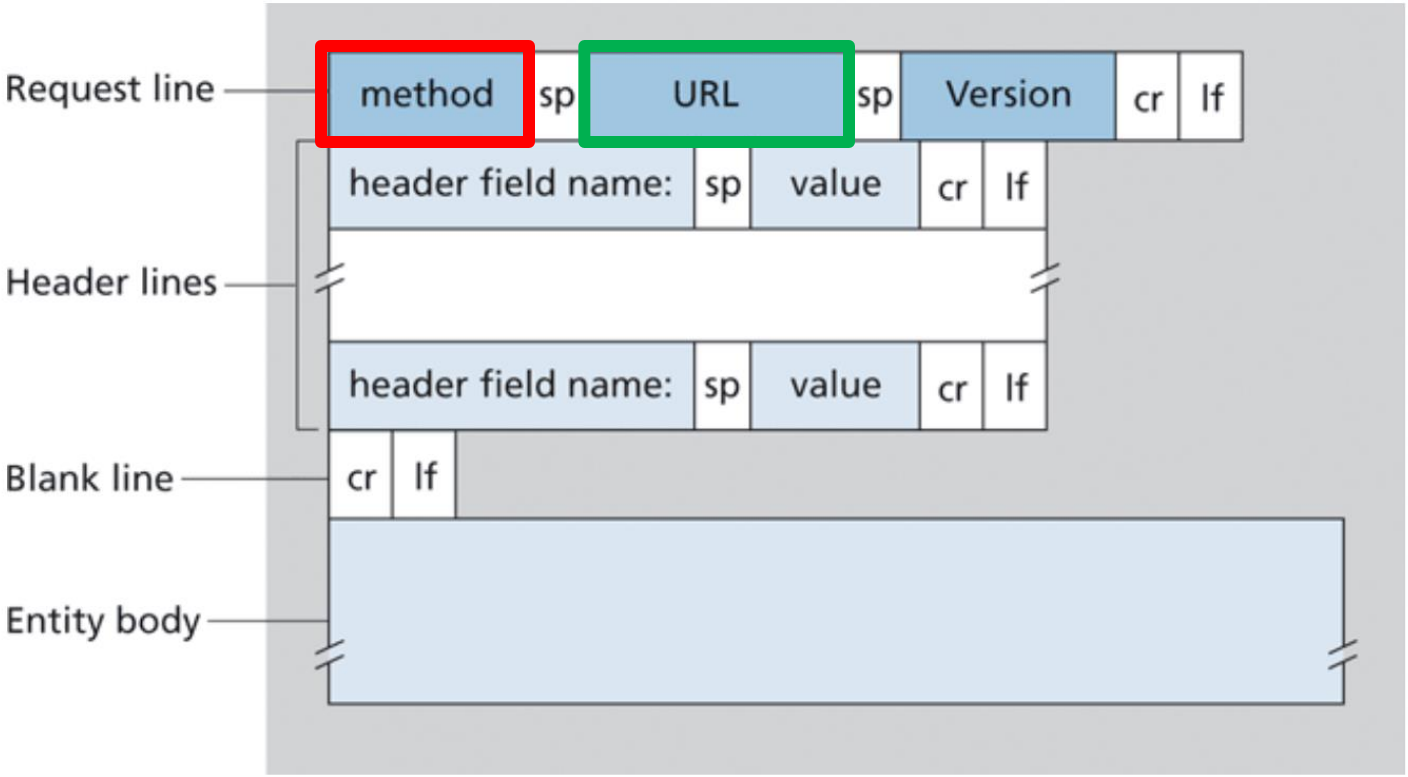


HTTP Request Message Format



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

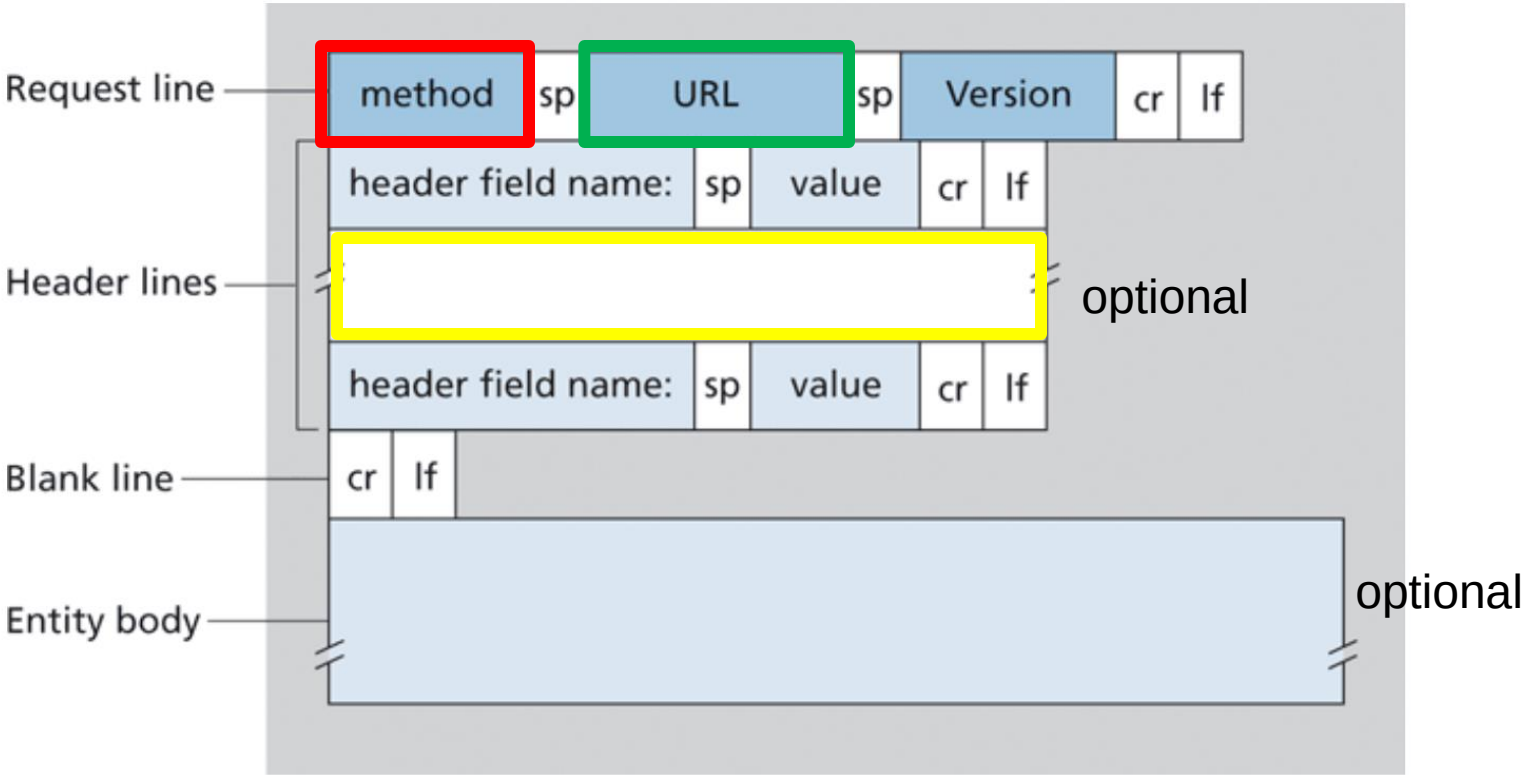
HTTP Request Message Format



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

`www.someschool.edu/meme.jpeg`

HTTP Request Message Format

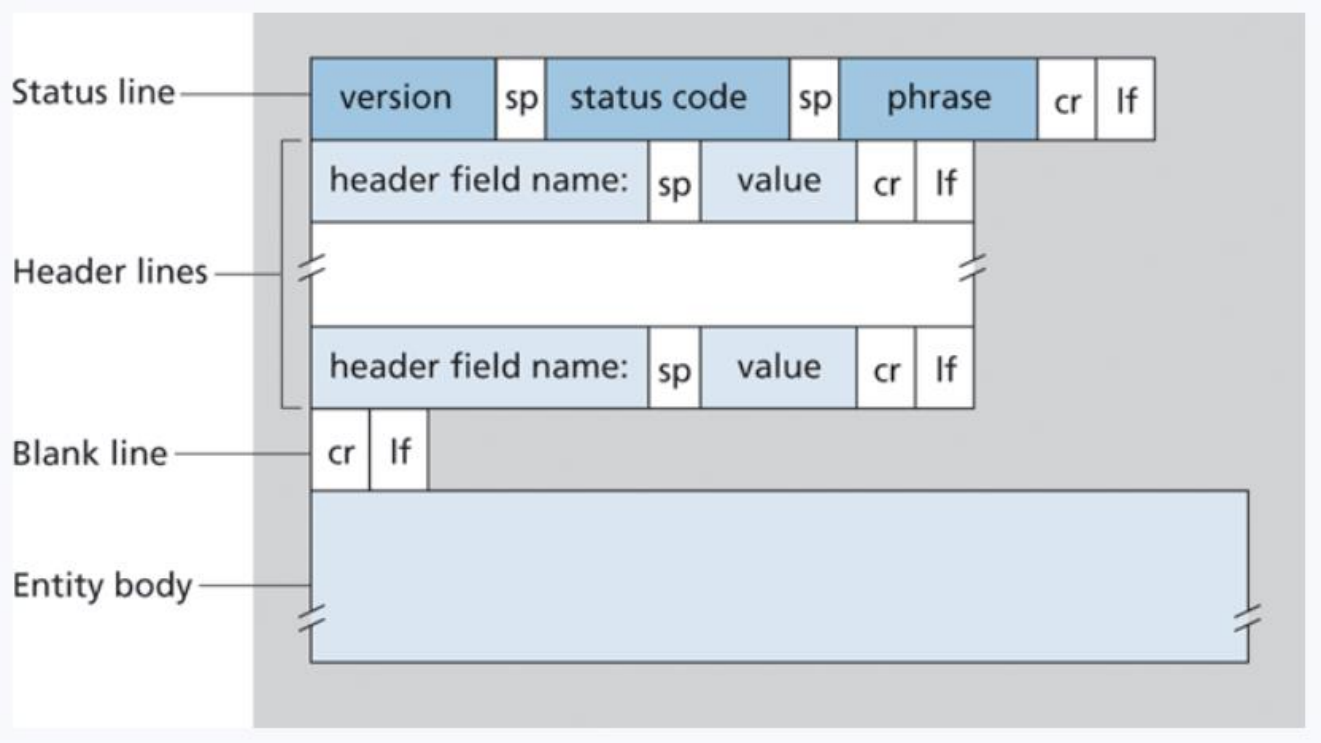


GET: Download resource
HEAD: Get resource metadata
POST: Upload form contents
PUT: Upload object to URL
DELETE: Delete object from URL

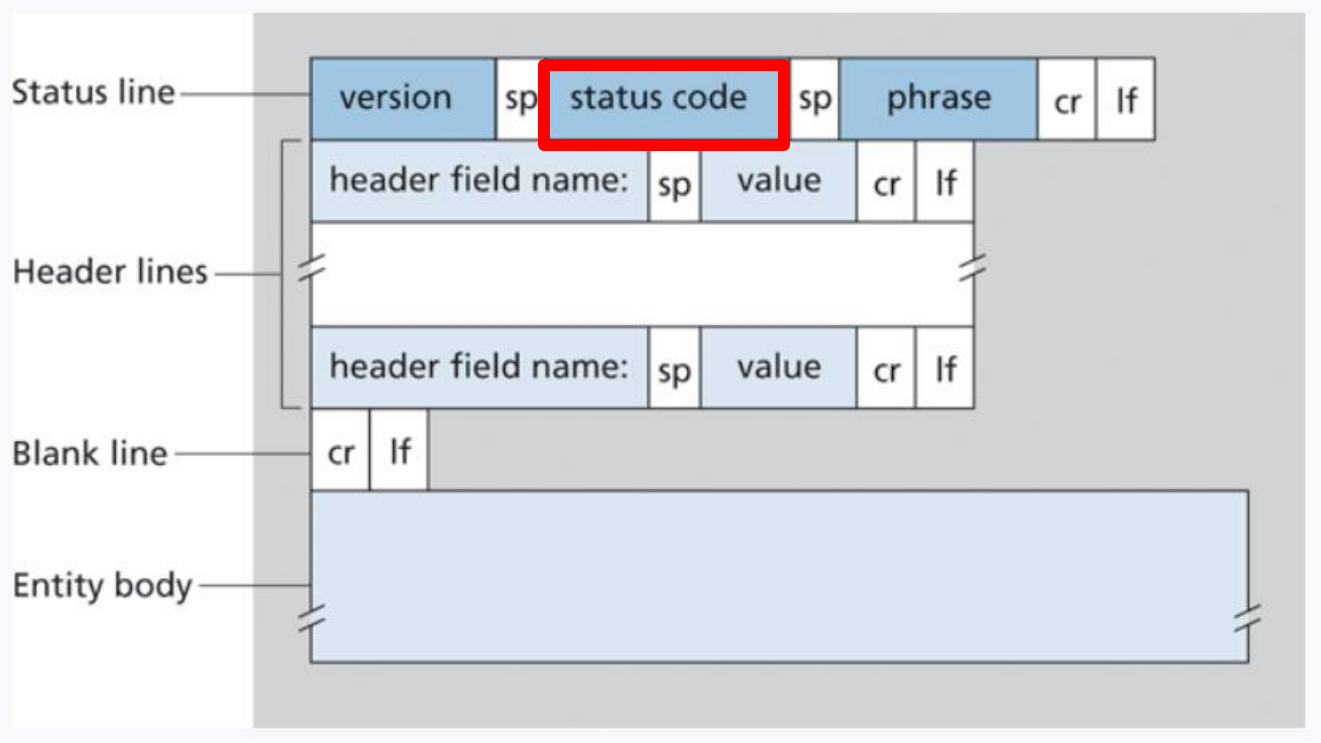
www.someschool.edu/meme.jpeg

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

HTTP Response Message Format

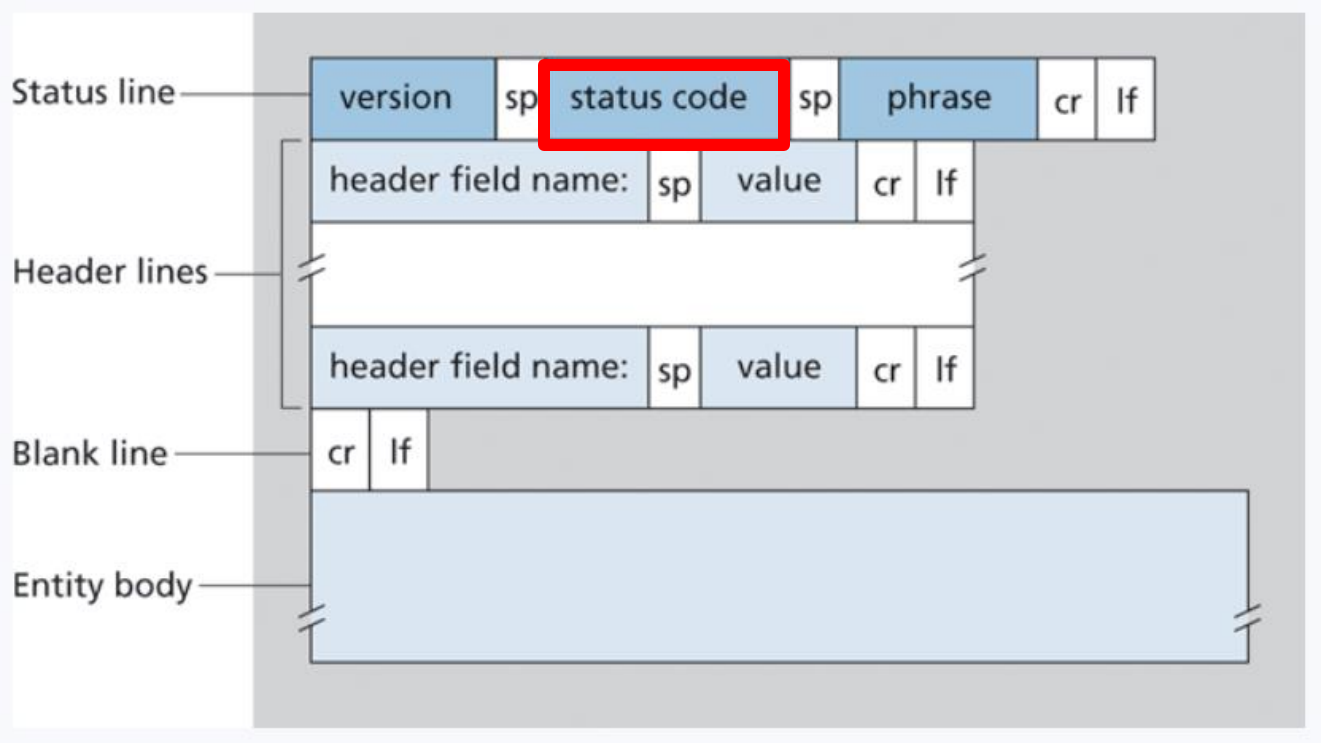


HTTP Response Message Format



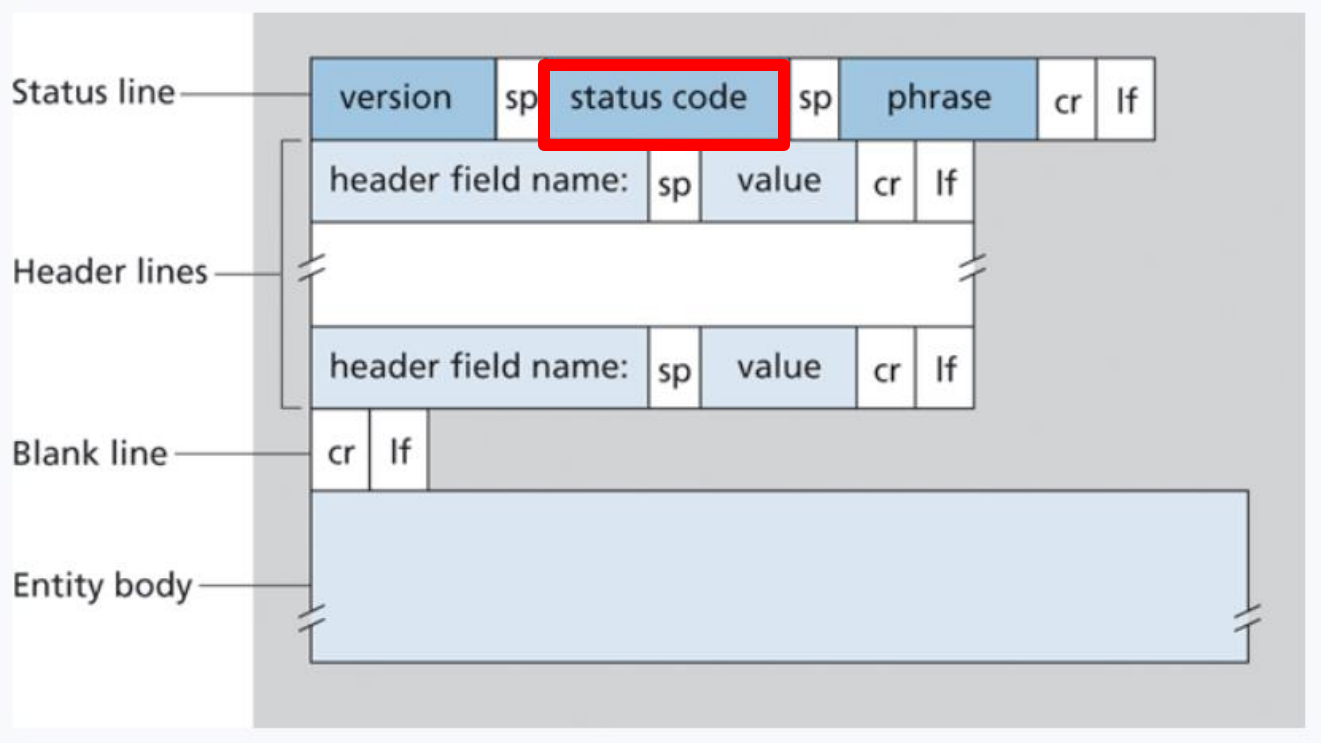
200

HTTP Response Message Format



200- Ok

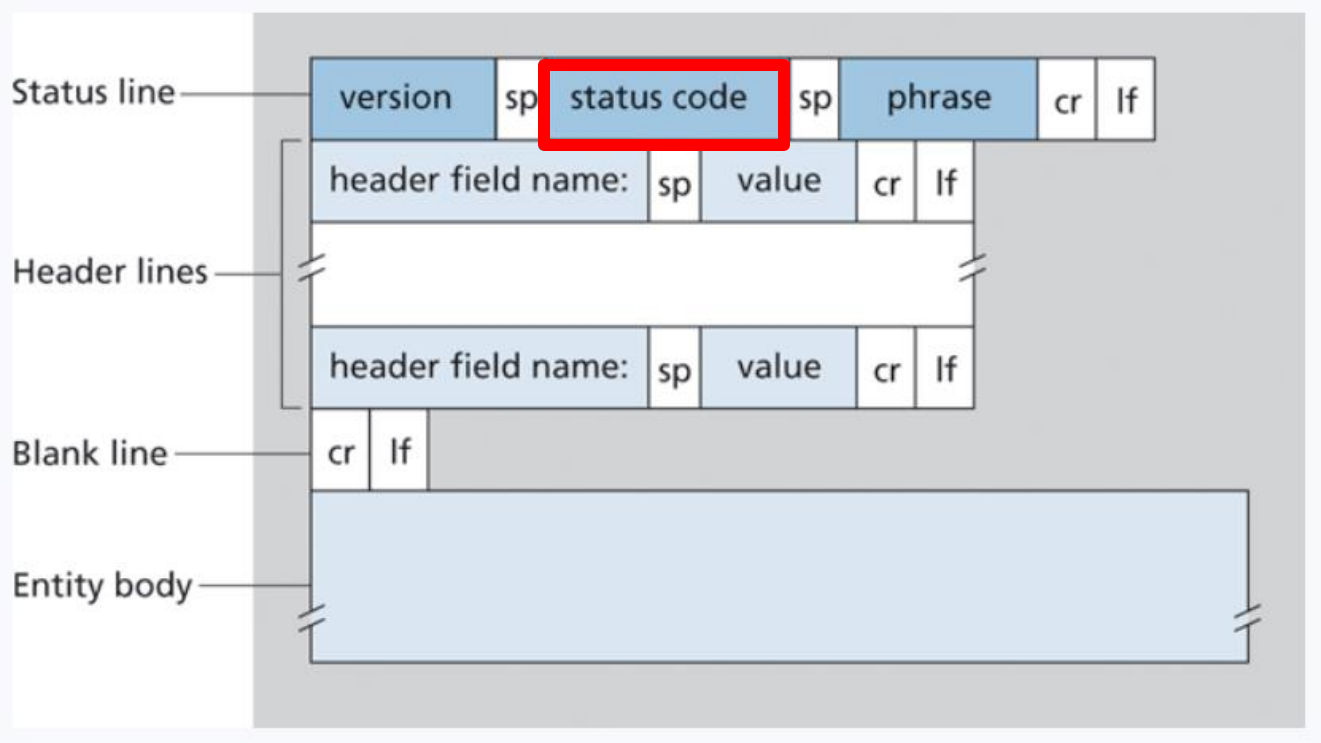
HTTP Response Message Format



200- Ok

404- Not found

HTTP Response Message Format

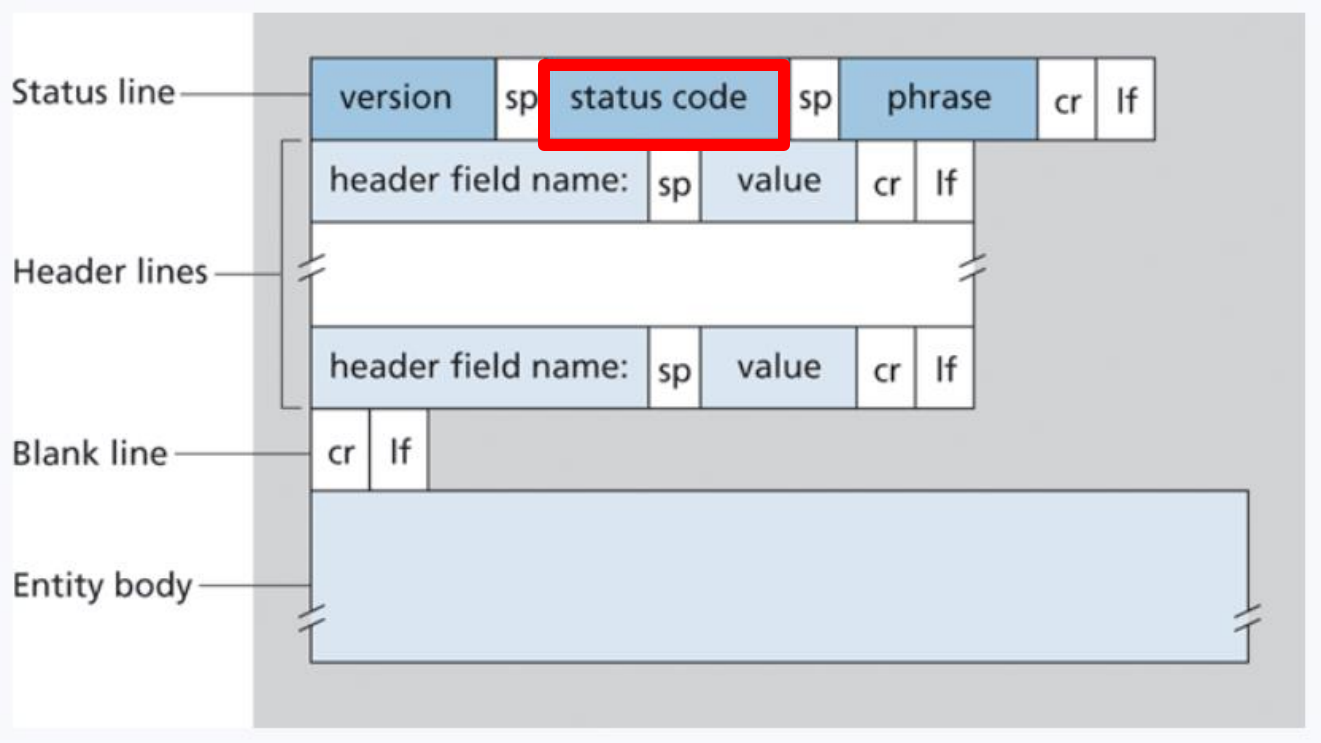


200- Ok

404- Not found

301- Resource has moved

HTTP Response Message Format



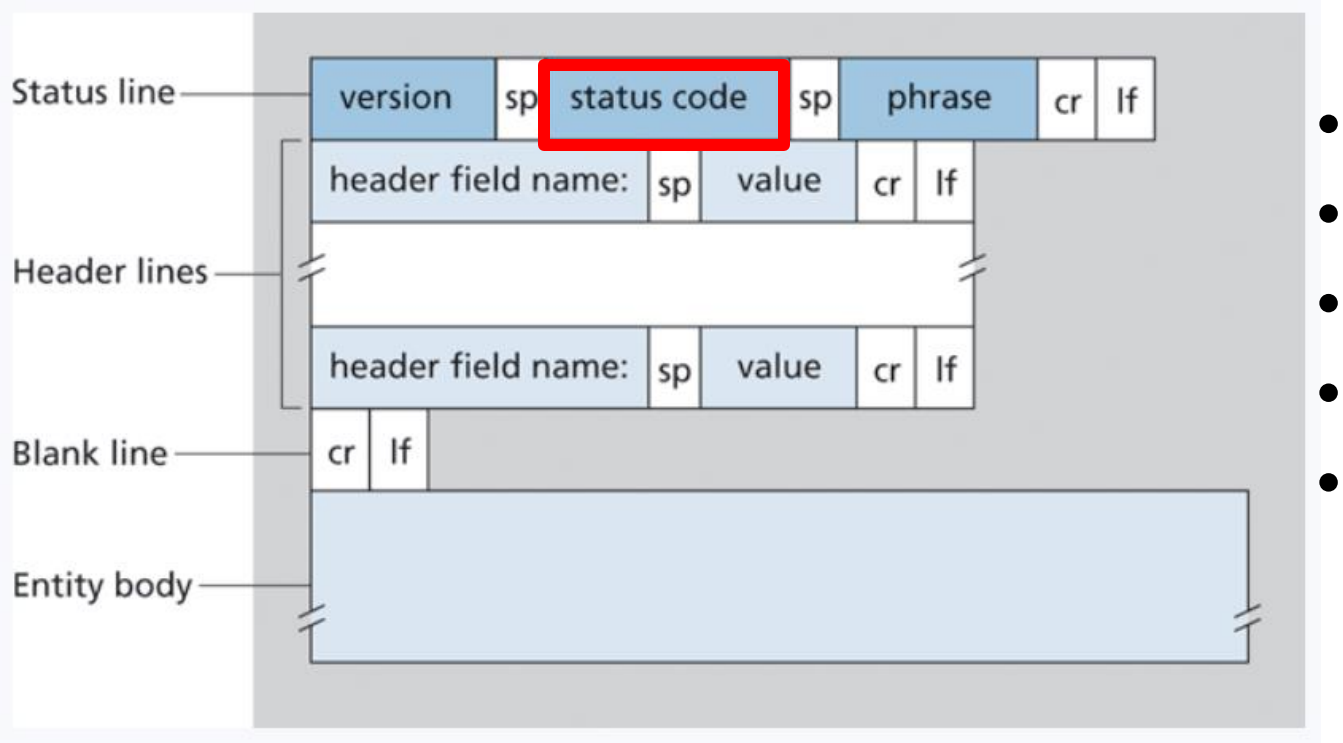
200- Ok

404- Not found

301- Resource has moved

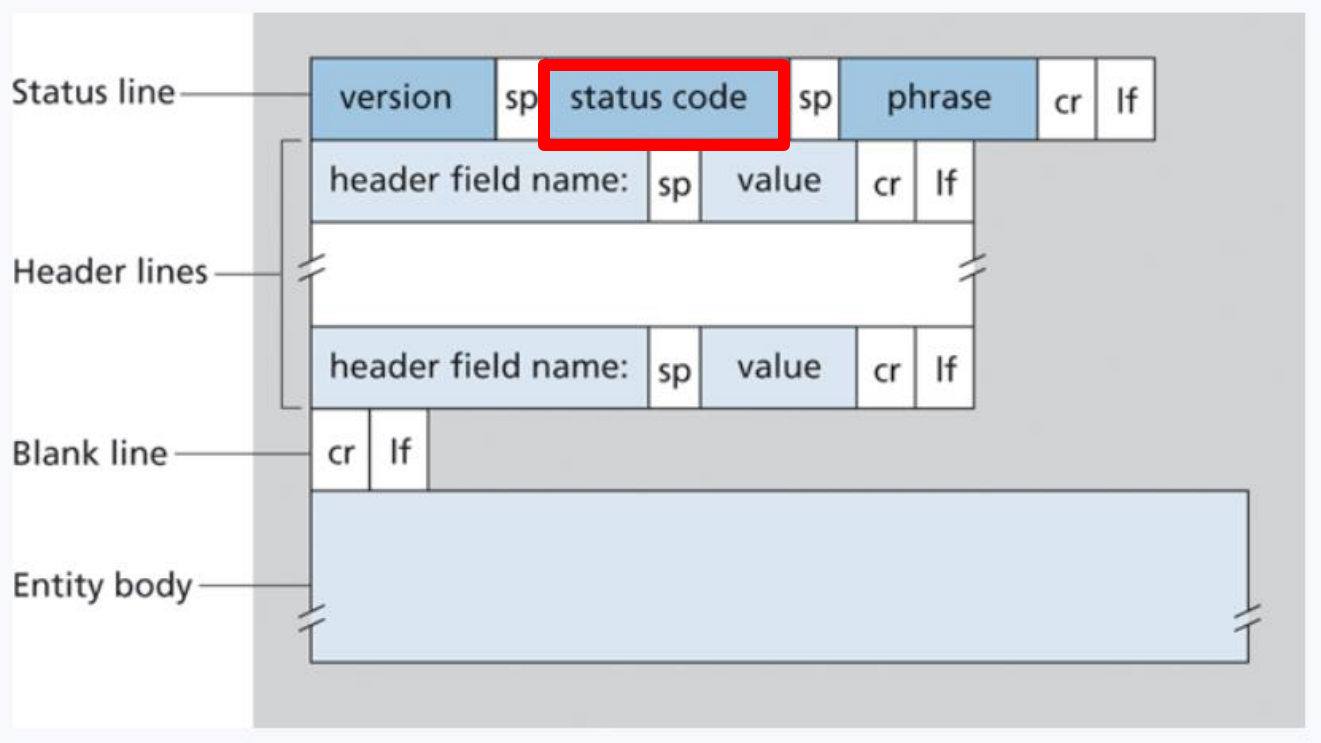
500- Internal server error

HTTP Response Message Format



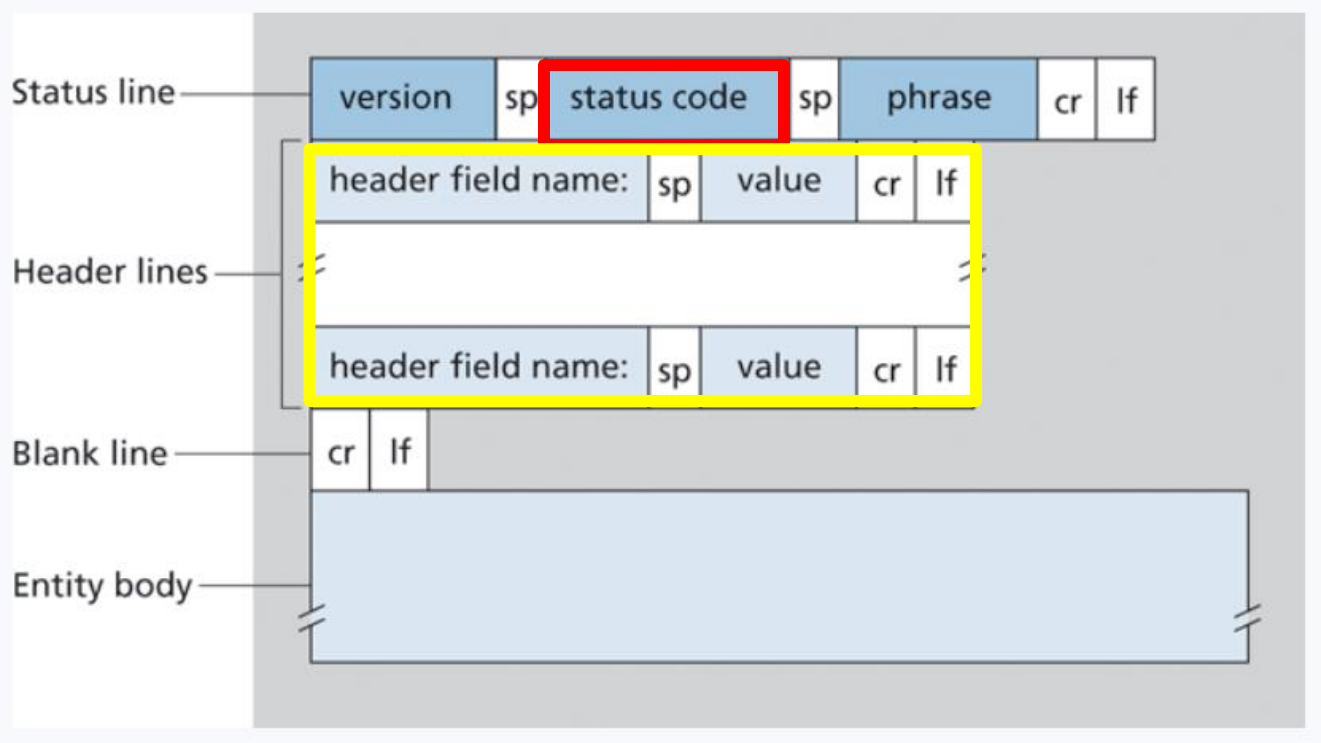
- Informational Responses (100s)
- Successful Responses (200s)
- Redirection messages (300s)
- Client error response (400s)
- Server error response (500s)

HTTP Response Message Format



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

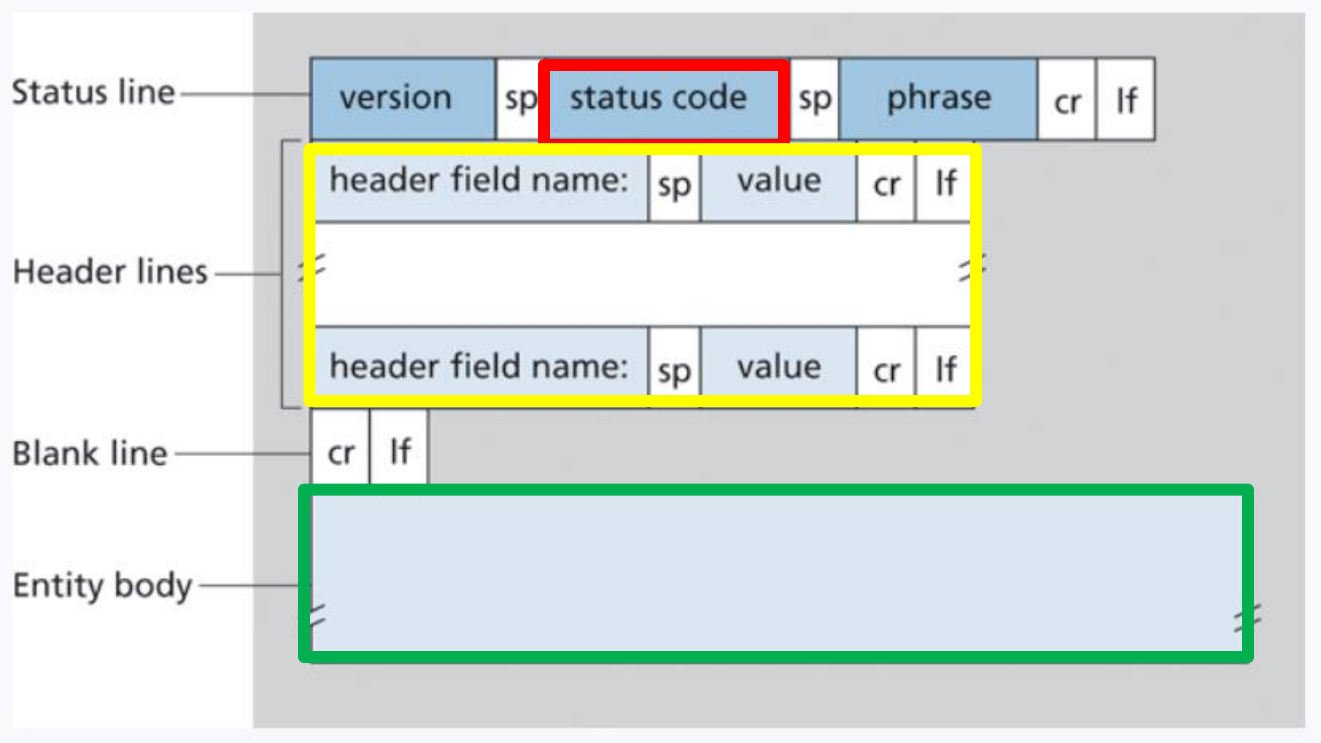
HTTP Response Message Format



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

```
Content-type: text/html
Content-length: 6821
Last-modified: …
```

HTTP Response Message Format



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

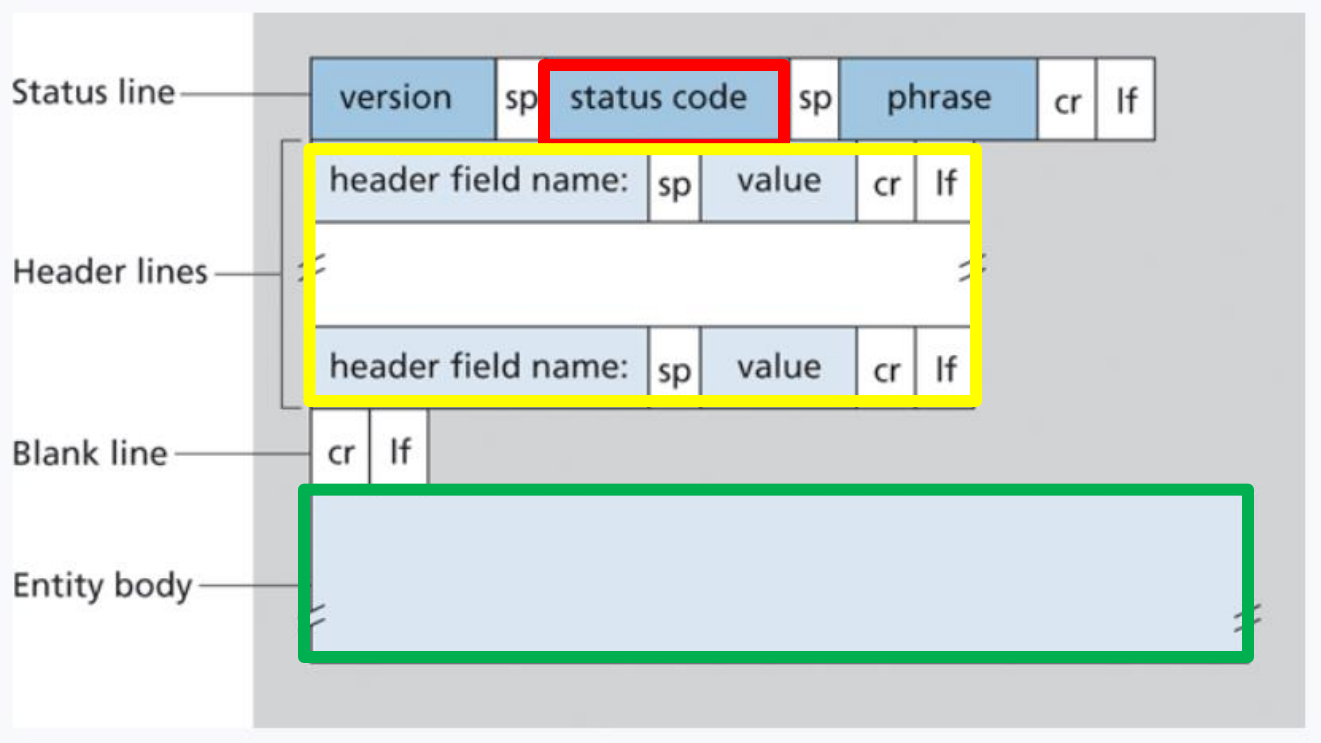
```
Content-type: text/html
Content-length: 6821
Last-modified: …
```

```
<!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 Response Message Format




```
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
```

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

```
Content-type: text/html
Content-length: 6821
Last-modified: …
```

```
<!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 in action