

# Mobile Application Development

Higher Diploma in Science in Computer Science

---

Produced  
by

Eamonn de Leastar (edeleastar@wit.ie)

Department of Computing, Maths & Physics  
Waterford Institute of Technology

<http://www.wit.ie>

<http://elearning.wit.ie>



Waterford Institute of Technology  
INSTITIÚID TEICNEOLAÍOCHTA PHORT LÁIRGE

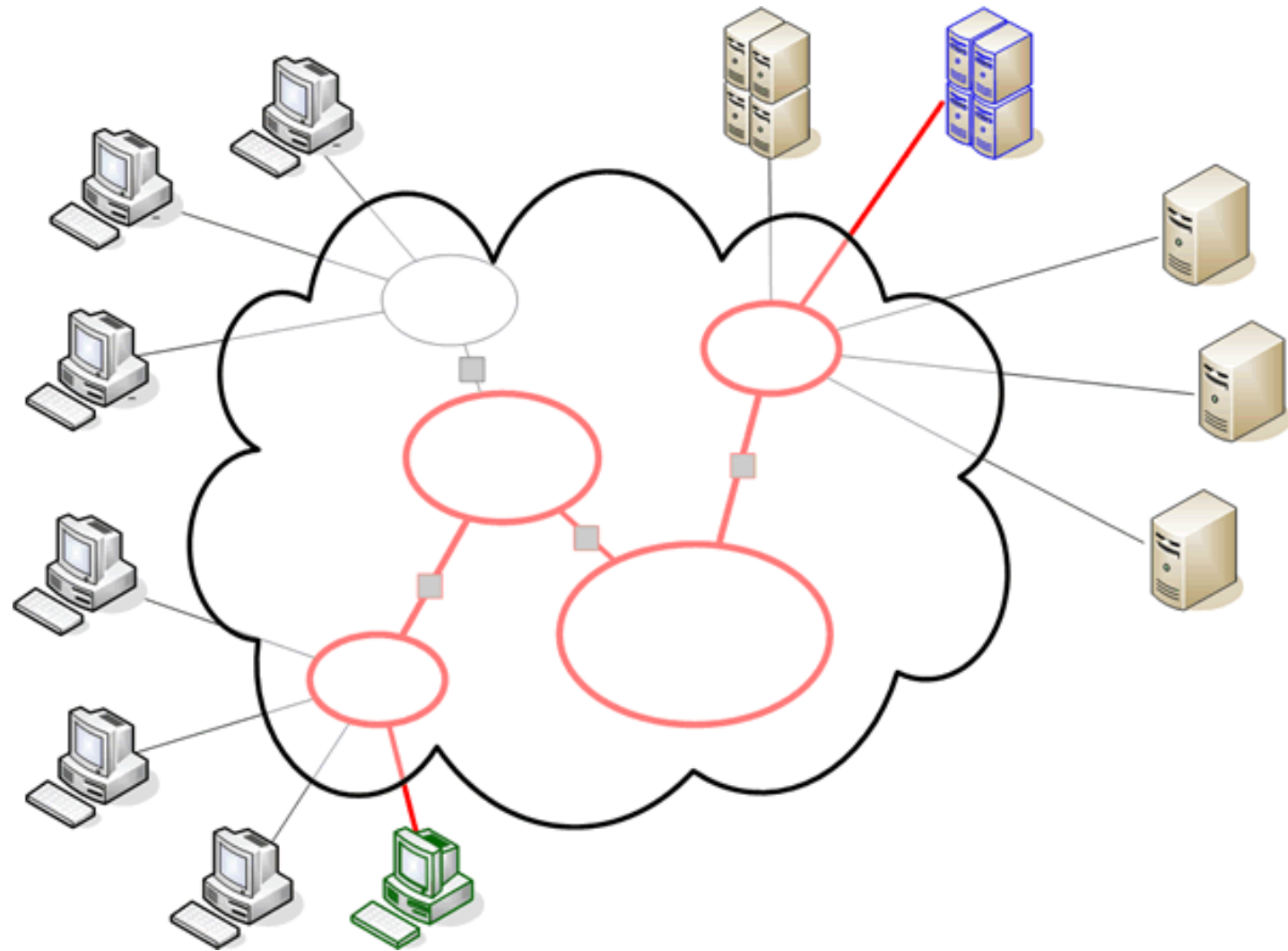
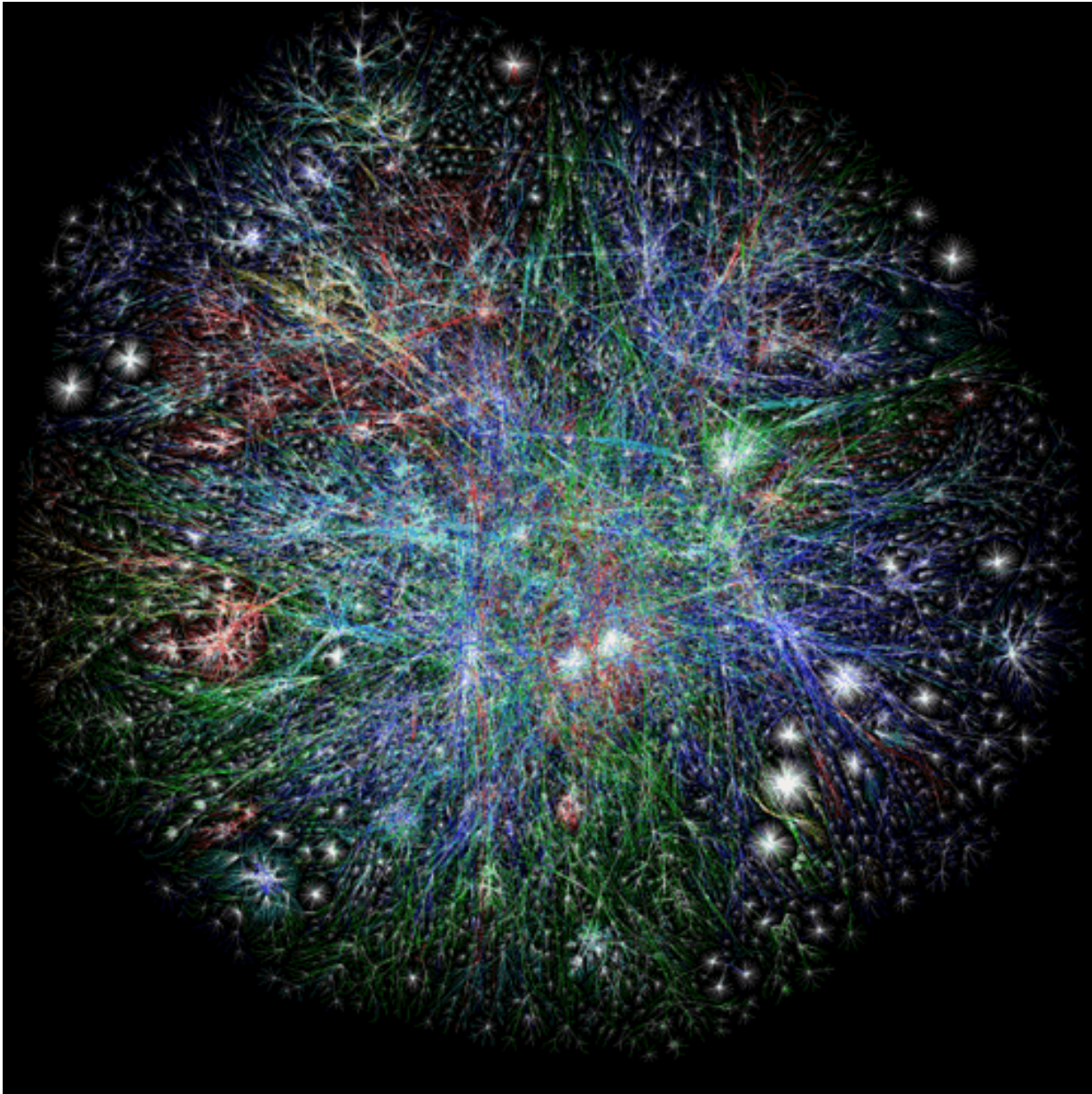


# OSI, the Internet & the HTTP Protocol

---

# The Internet

---



# Underlying nature of the Internet - Protocols & Standards

---

“The irony is that in all its various guises -- commerce, research, and surfing -- the Web is already so much a part of our lives that familiarity has clouded our perception of the Web itself.”

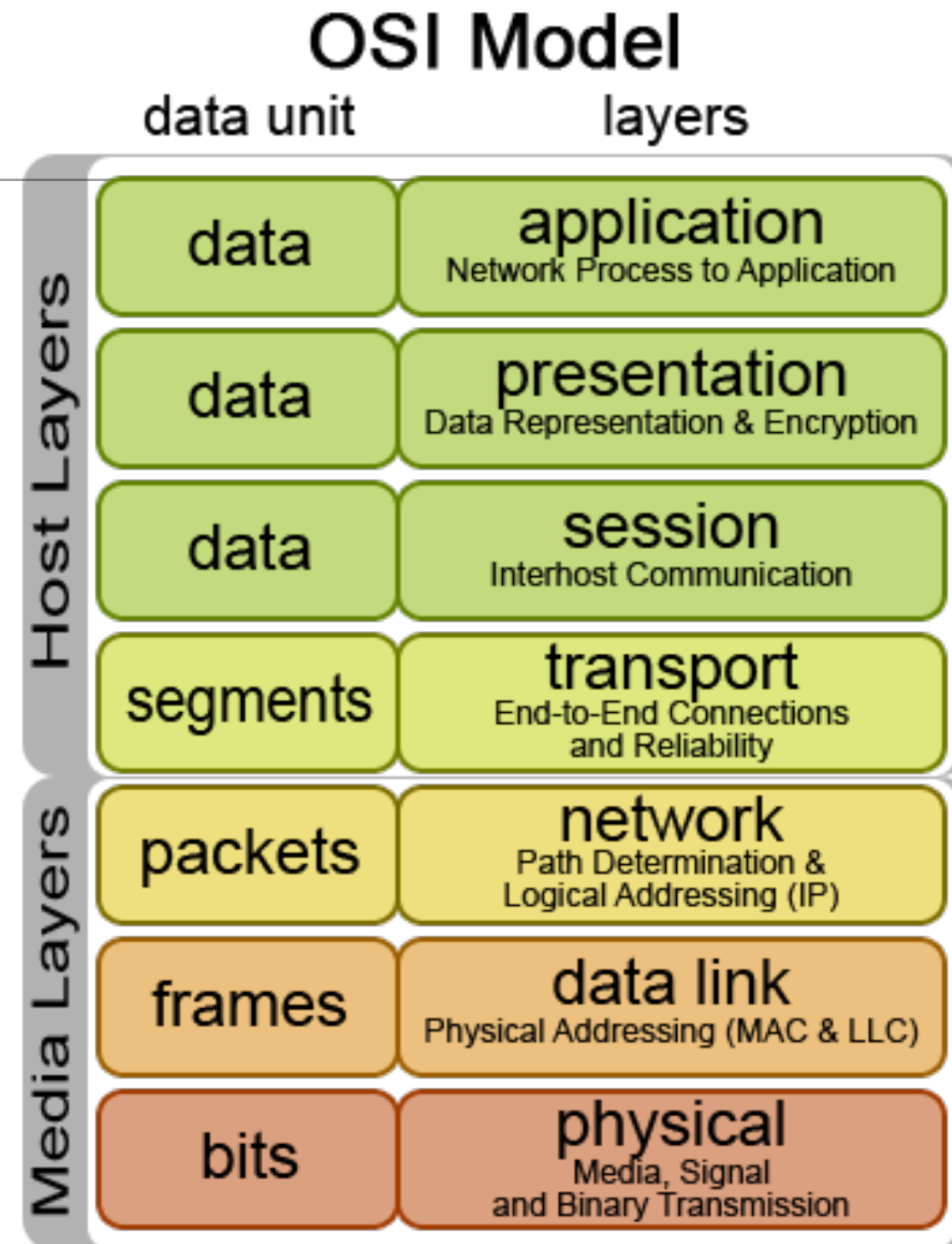
Tim Berners-Lee in Weaving the Web



- Email
  - SMTP
  - POP
  - IMAP
- File Transfer
  - FTP
  - SFTP
  - FTP over SSH
- Login
  - Telnet
  - SSH
- Web
  - HTTP, HTML
- Messaging
  - XMPP

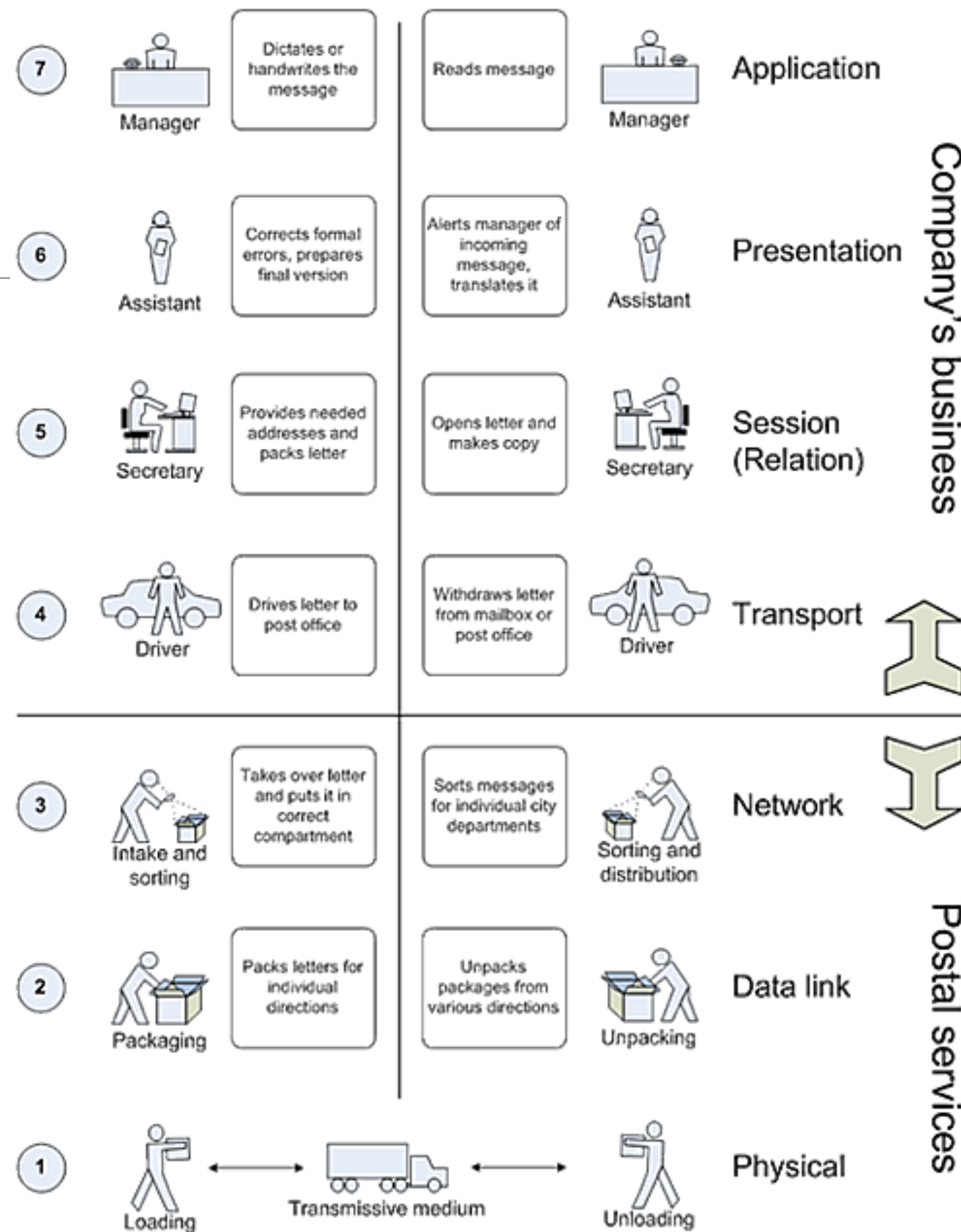
# OSI Model

- The OSI model divides the functions of a protocol into a series of layers - each layer only uses the functions of the layer below, and only exports functionality to the layer above.
- A system that implements protocol behavior consisting of a series of these layers is known as a 'protocol stack' or 'stack'
- The interface between layers dictates the specifications on how one layer interacts with another. These specifications are typically known as Requests for Comments or "RFC"s in the TCP/IP community.



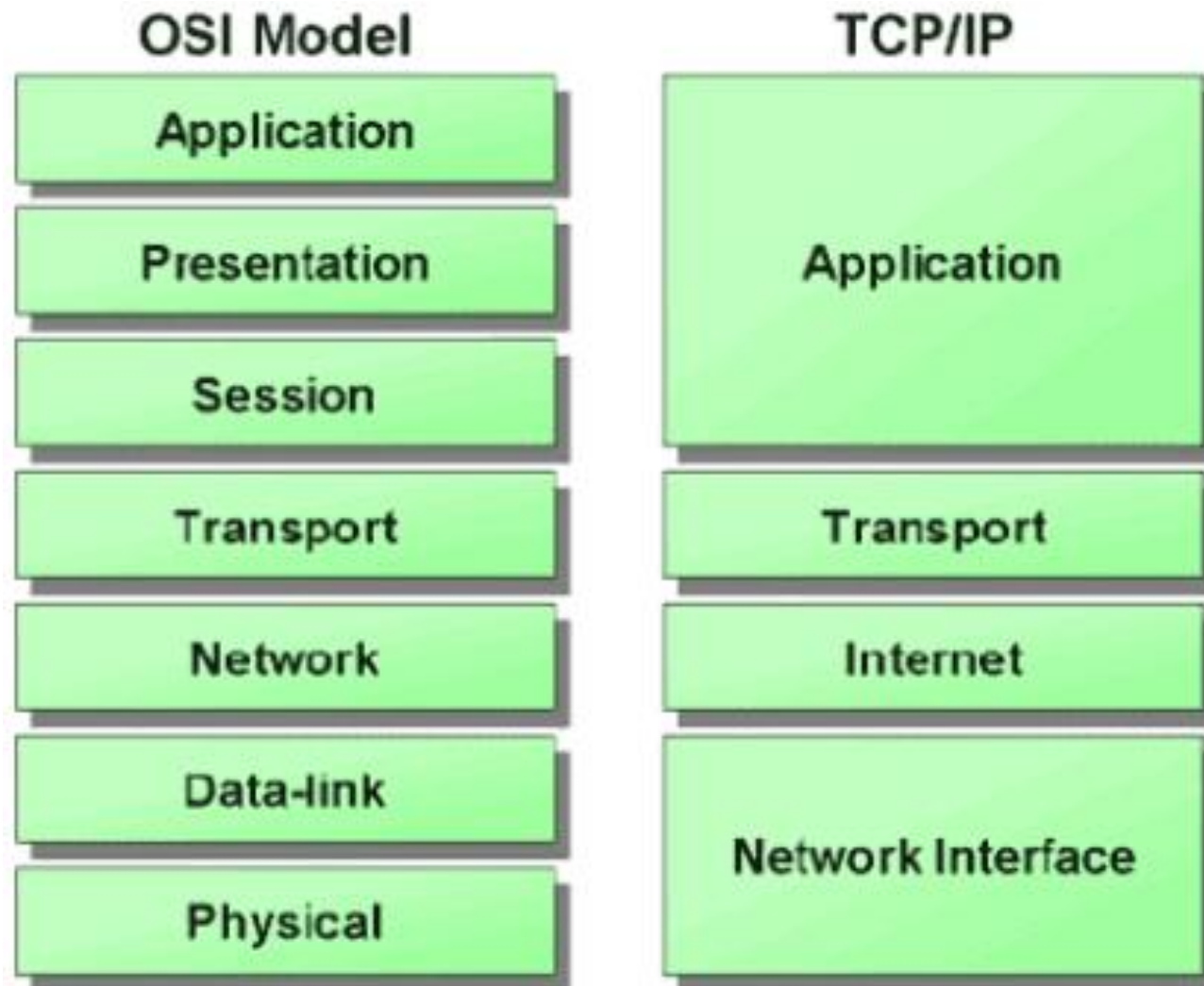


# OSI Analogy



RM – OSI and letter communication parallel

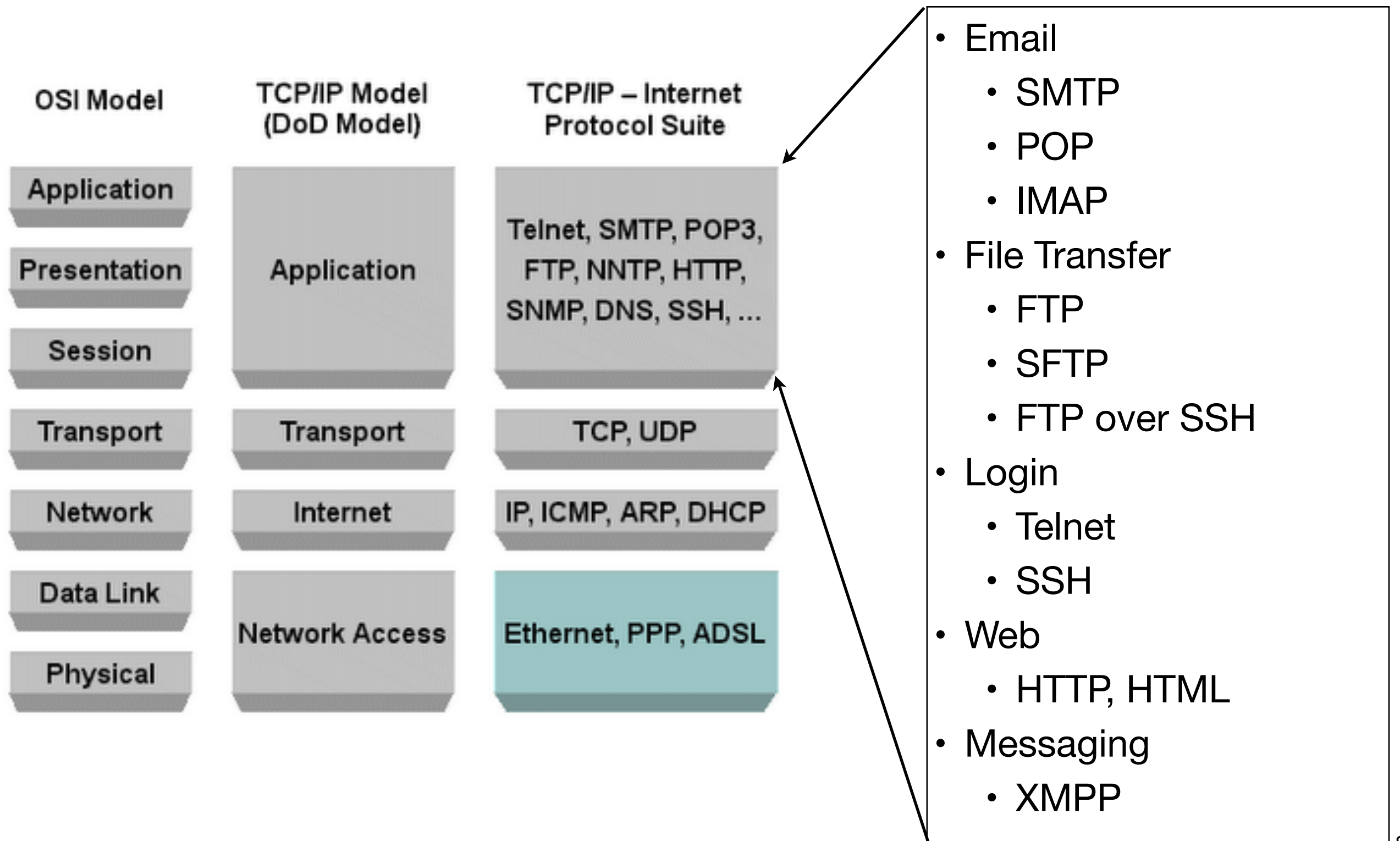
# OSI & TCP/IP



TCP/IP and the OSI model

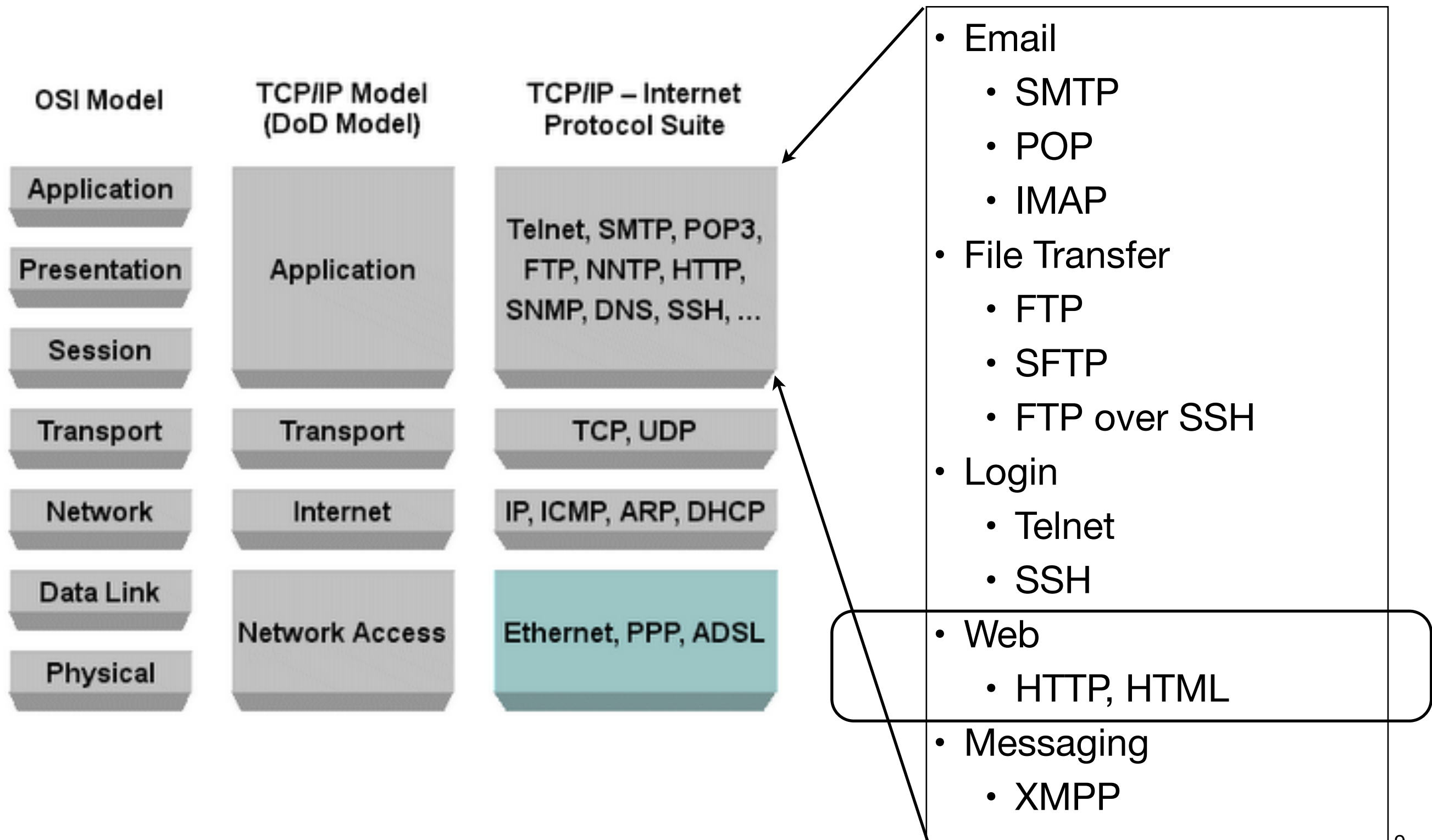
- The application-centric layers of the 7-layer model are condensed into a single application layer protocol in the TCP/IP stack
- the TCP/IP protocol stack defines a Network Interface instead of the Data-Link layer of the ISO model, and the Internet Layer of the TCP/IP mode is broadly equivalent to the Network Layer in the ISO model.

# TCP/IP Protocol Suite





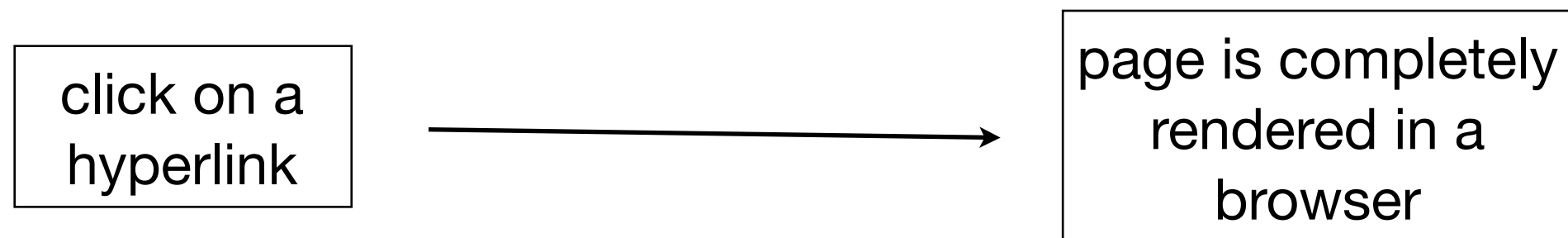
# TCP/IP Protocol Suite



# Lifecycle of a Hyperlink

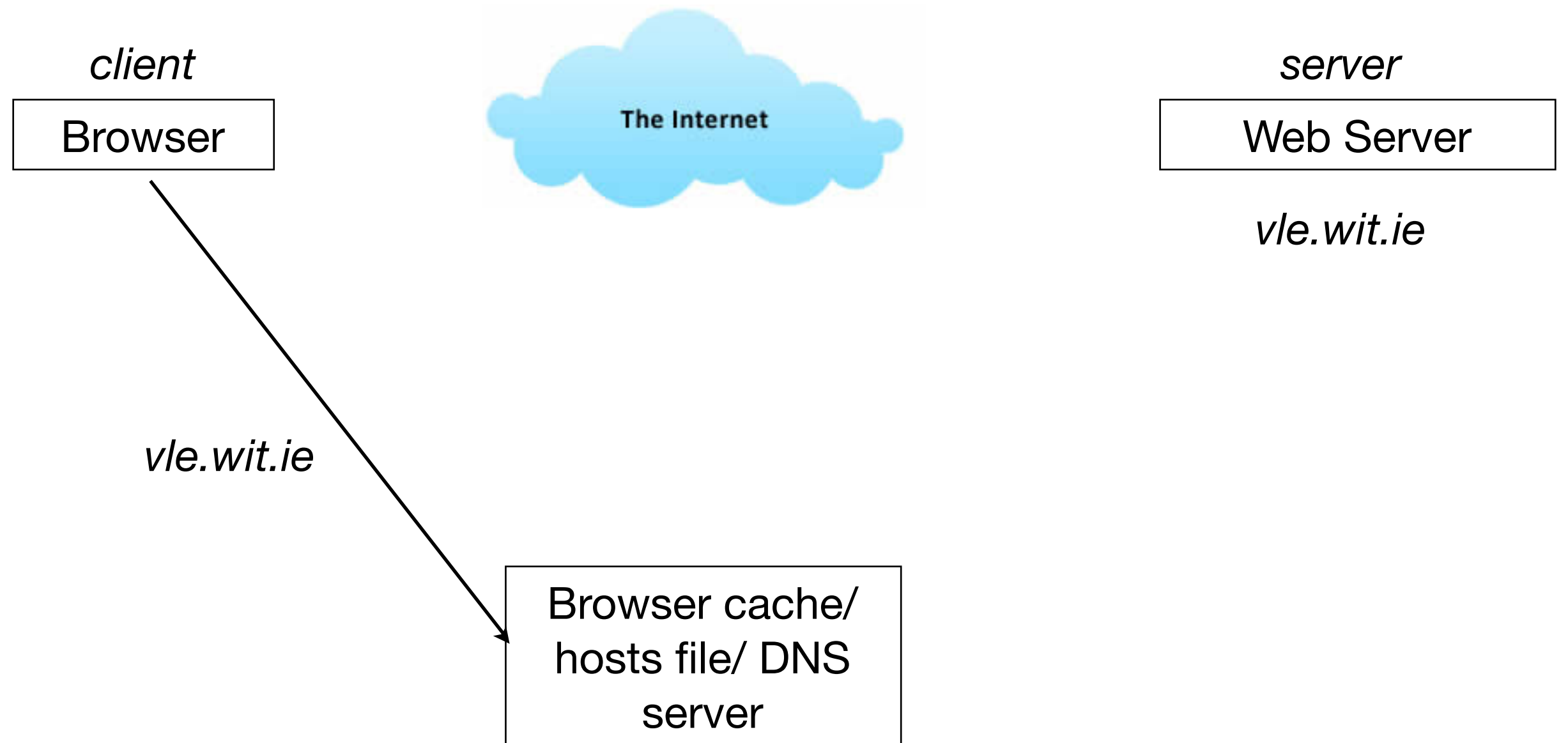
---

- What happens when:

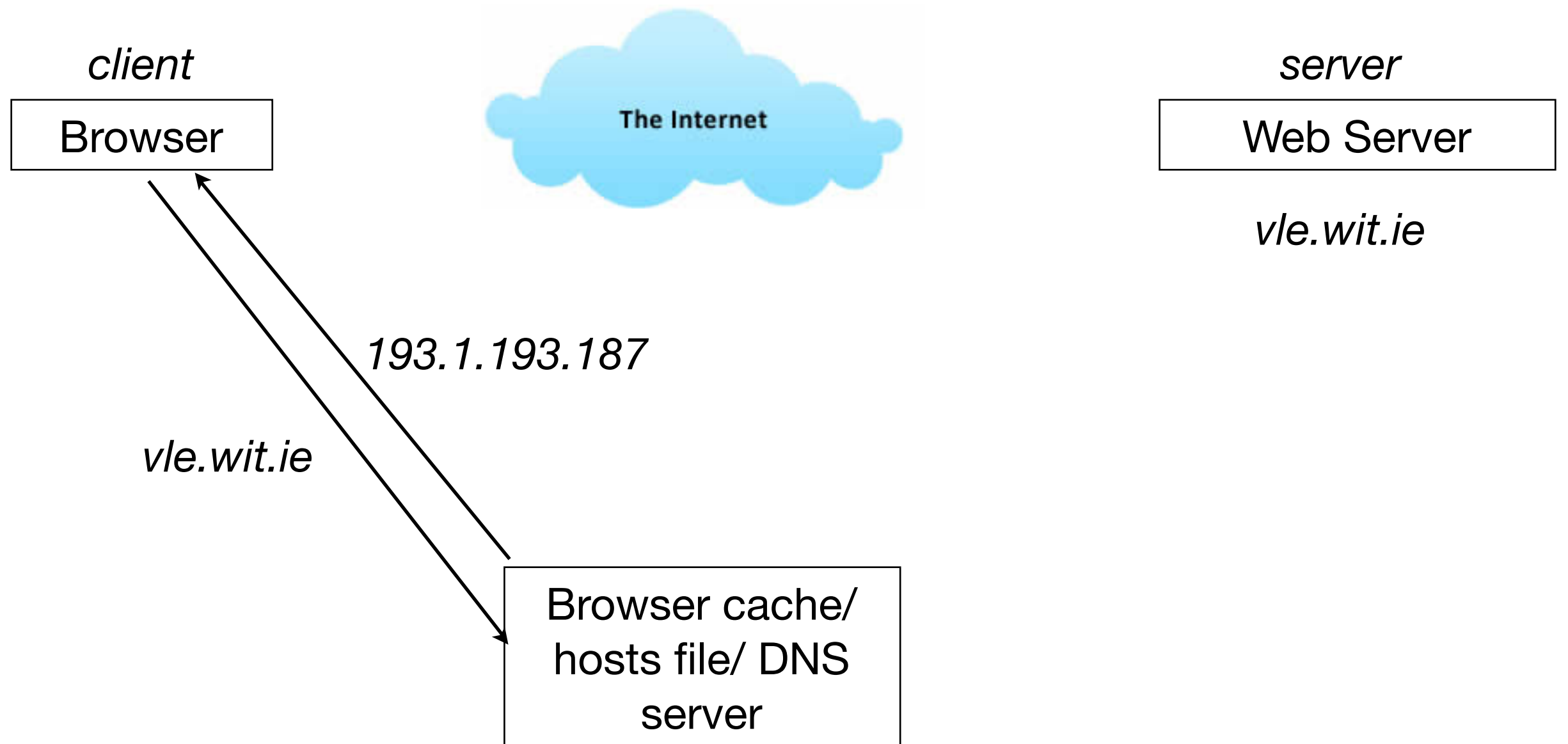


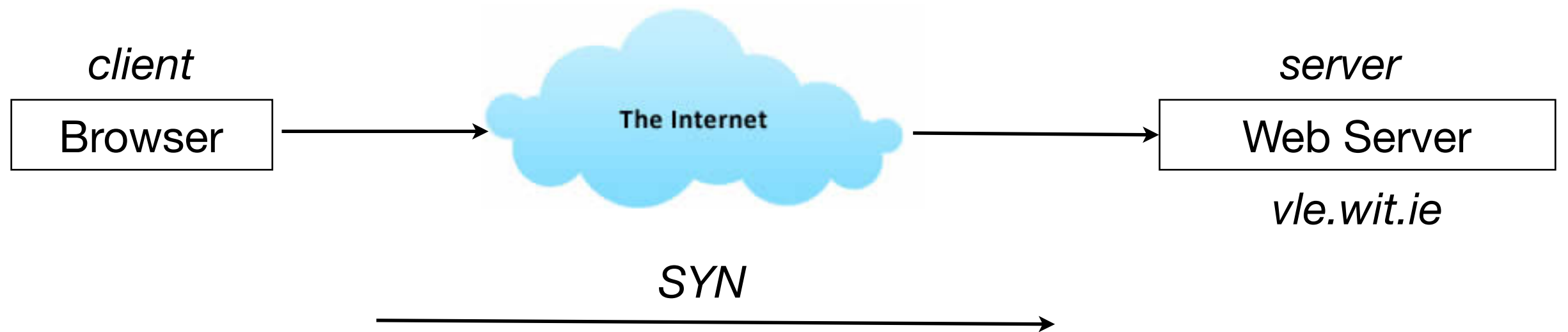




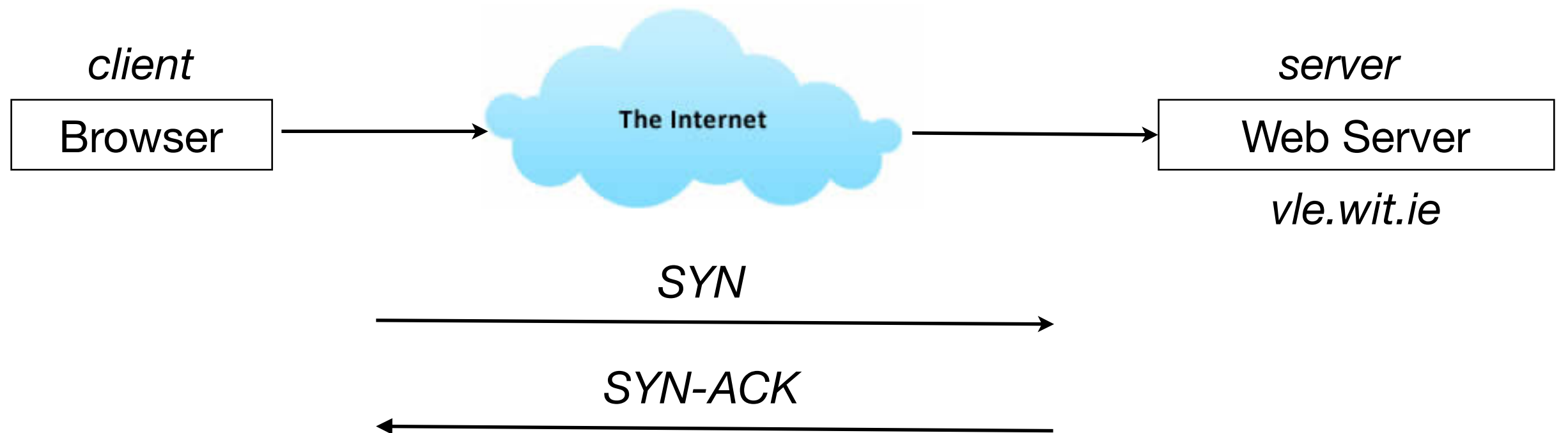




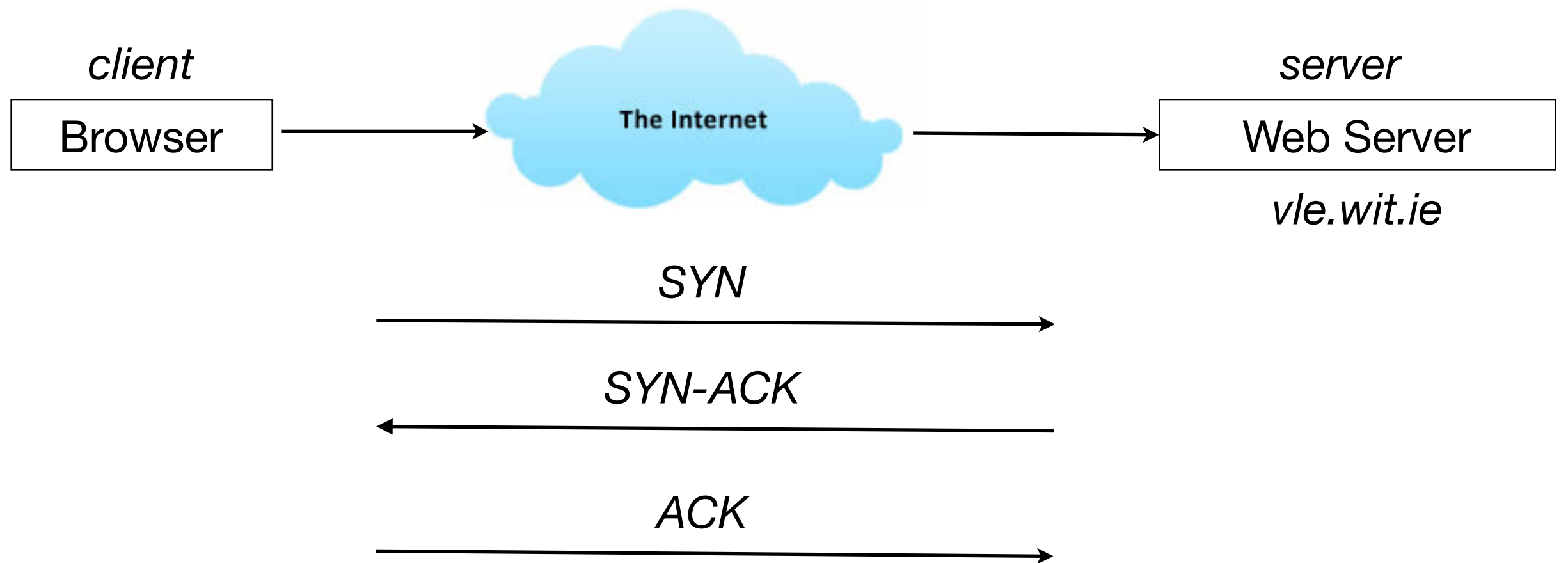




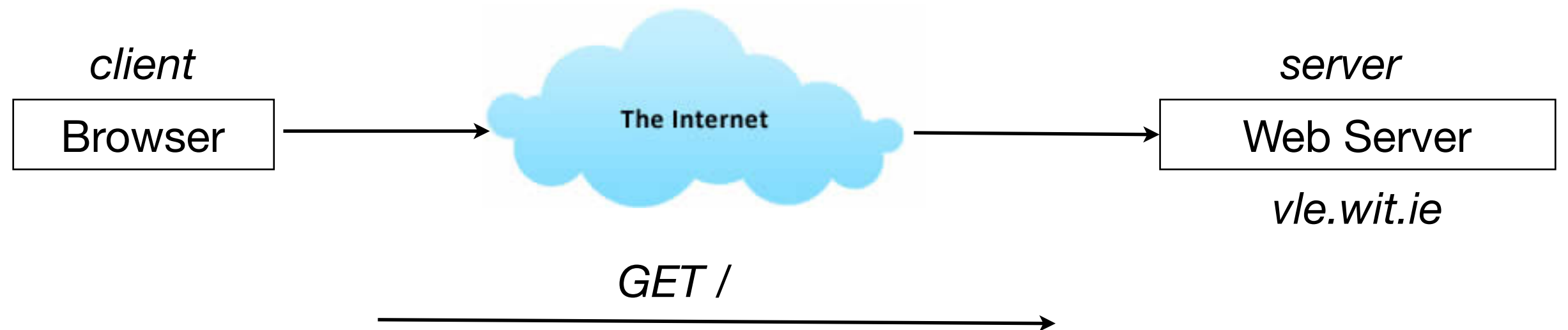
TCP: Are you there?



TCP: I'm here

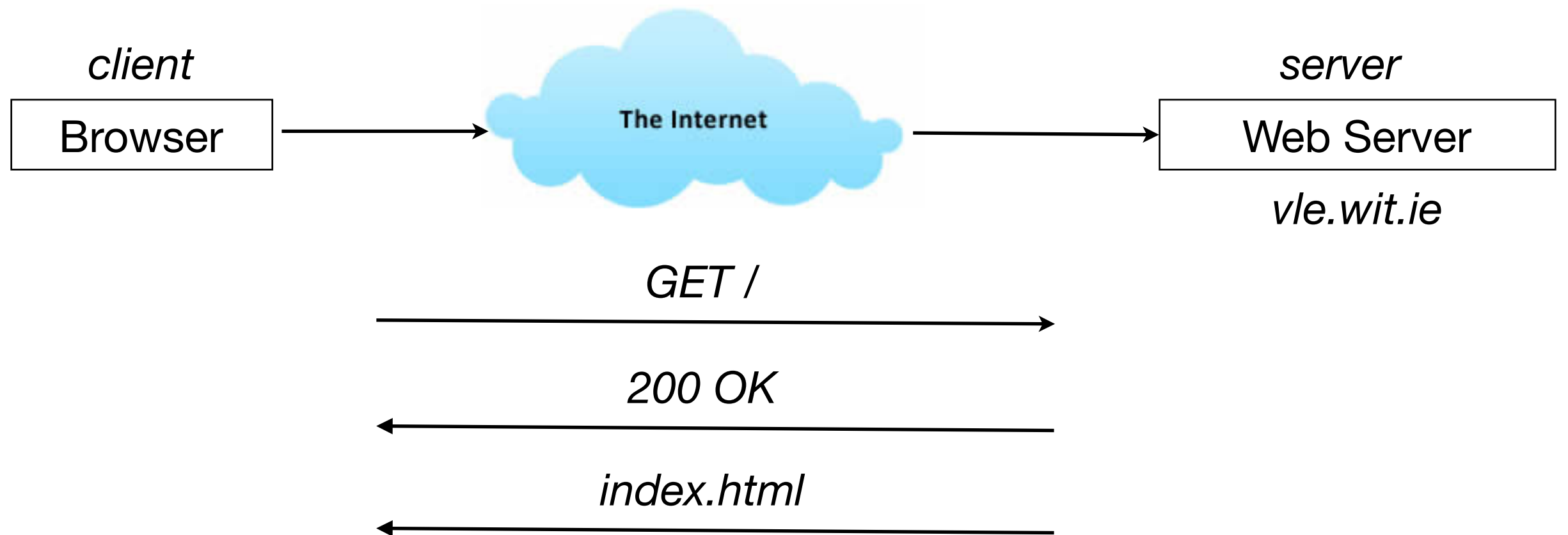


TCP: Acknowledged

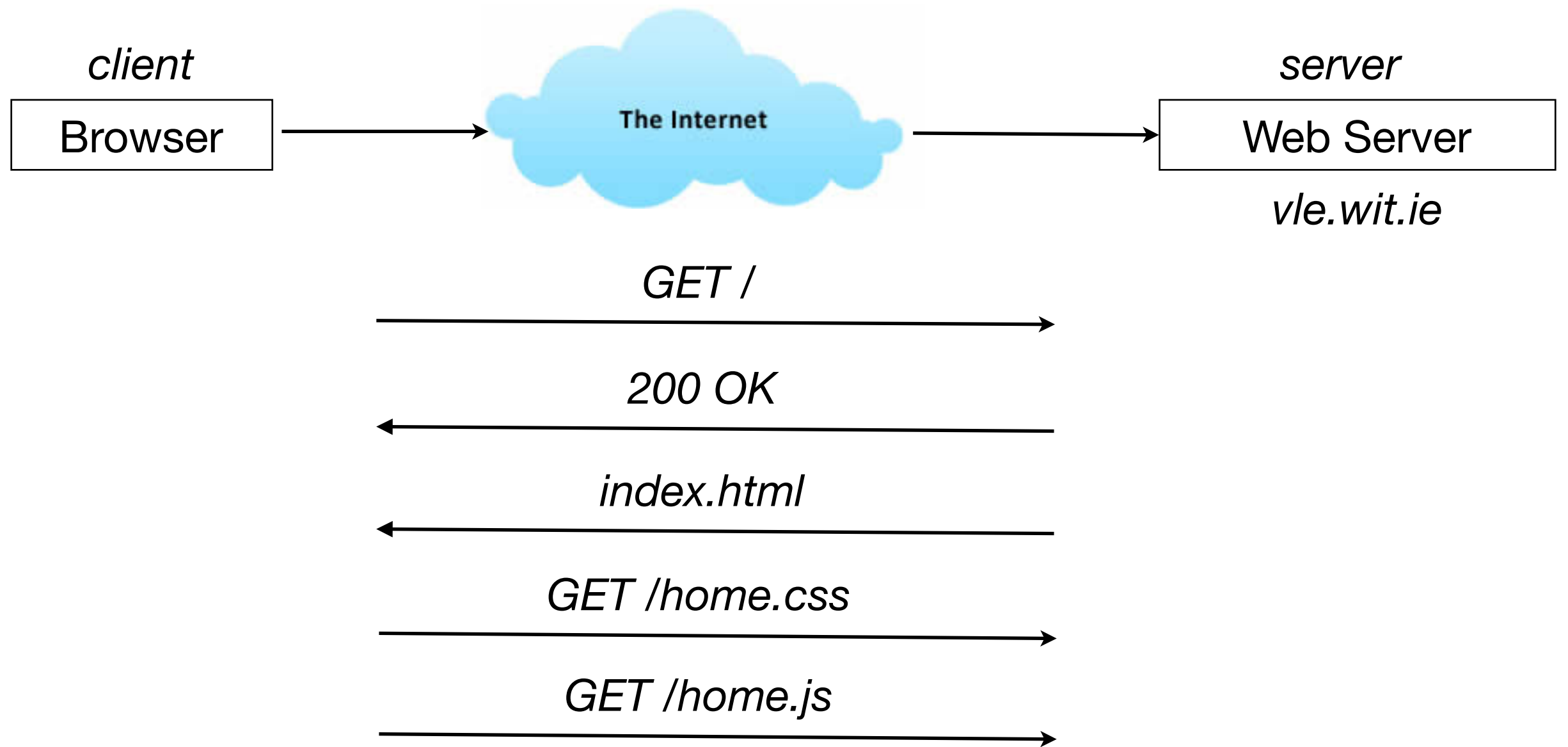


HTTP: Got this file?

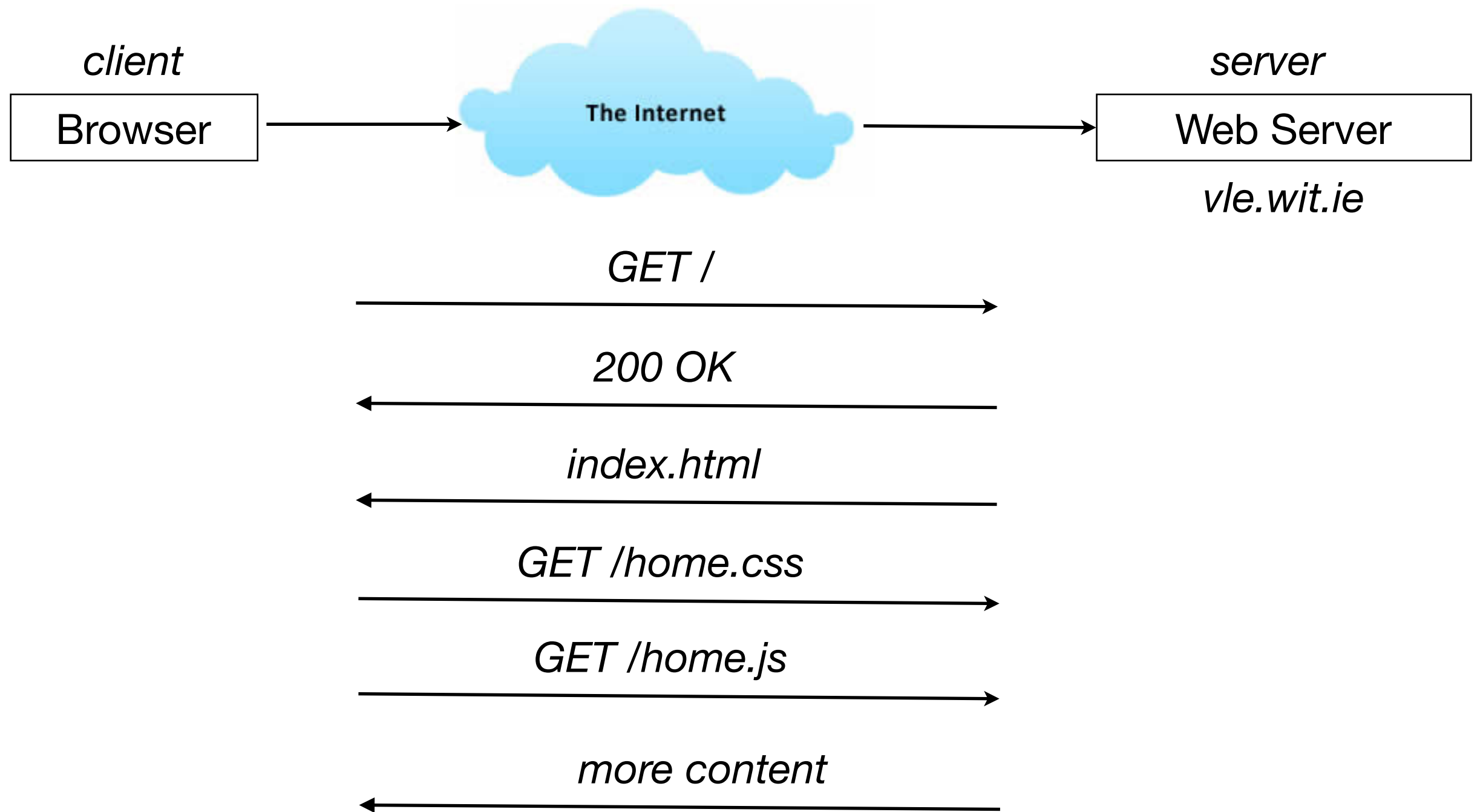




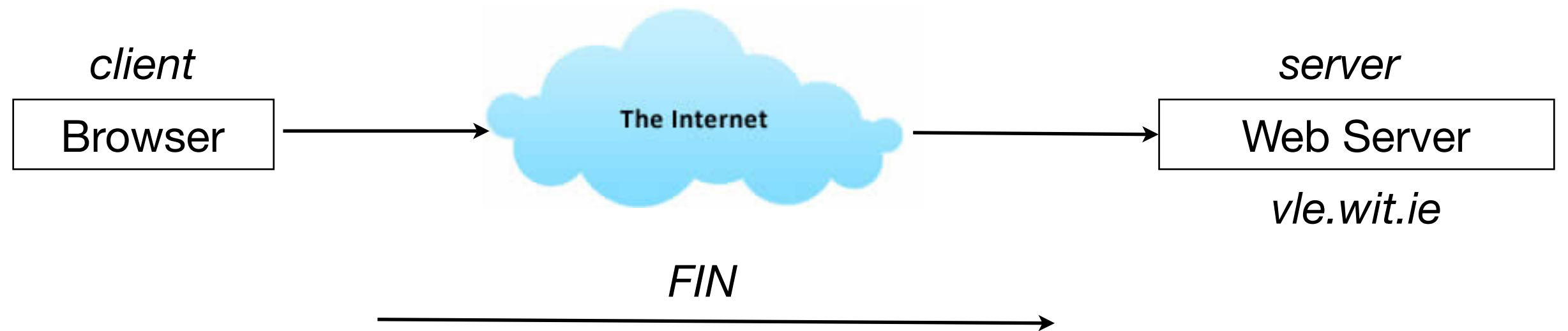
HTTP: Yes, here you go



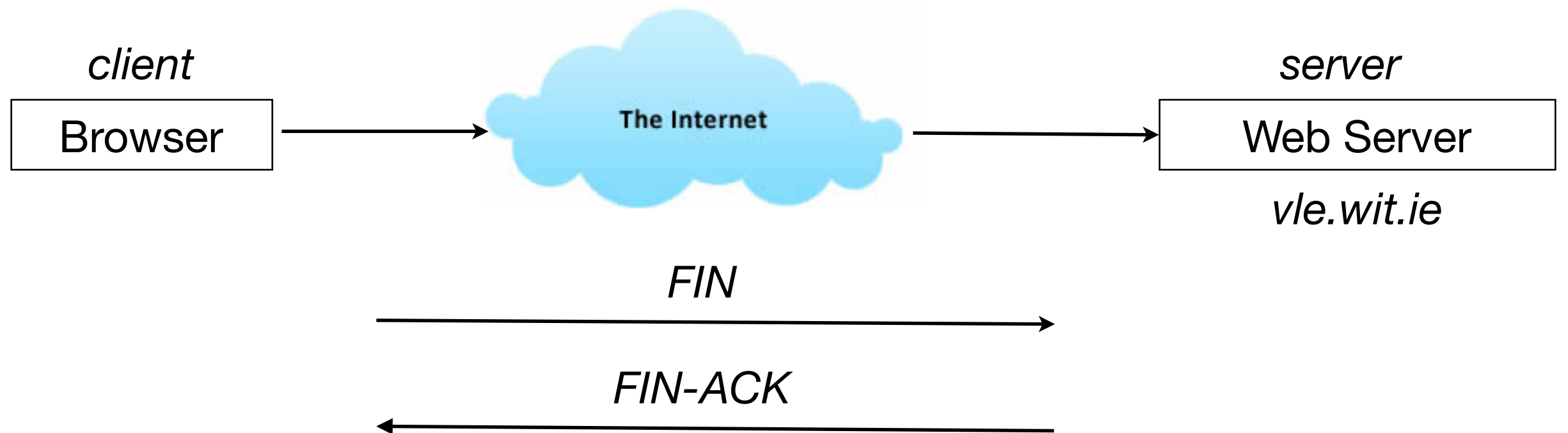
HTTP: And these as well?



HTTP: here you go



TCP: Finished

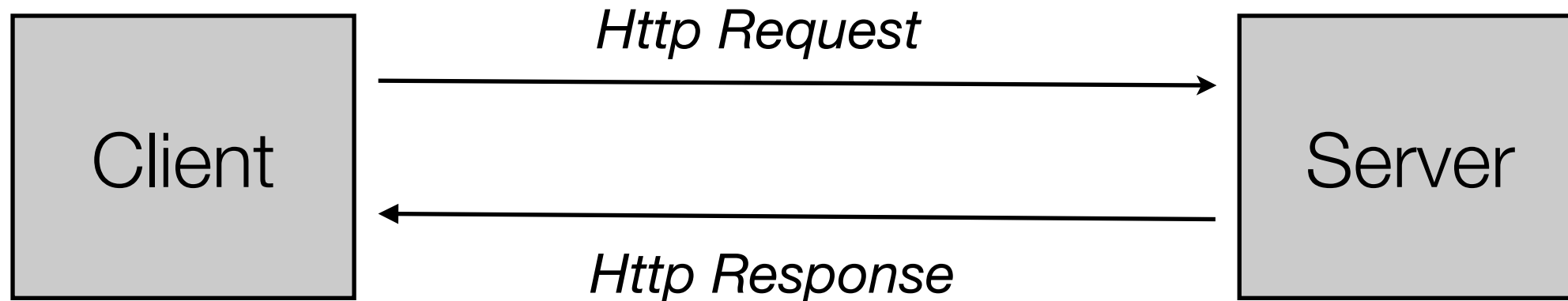


TCP: Acknowledged



# Client / Server

---

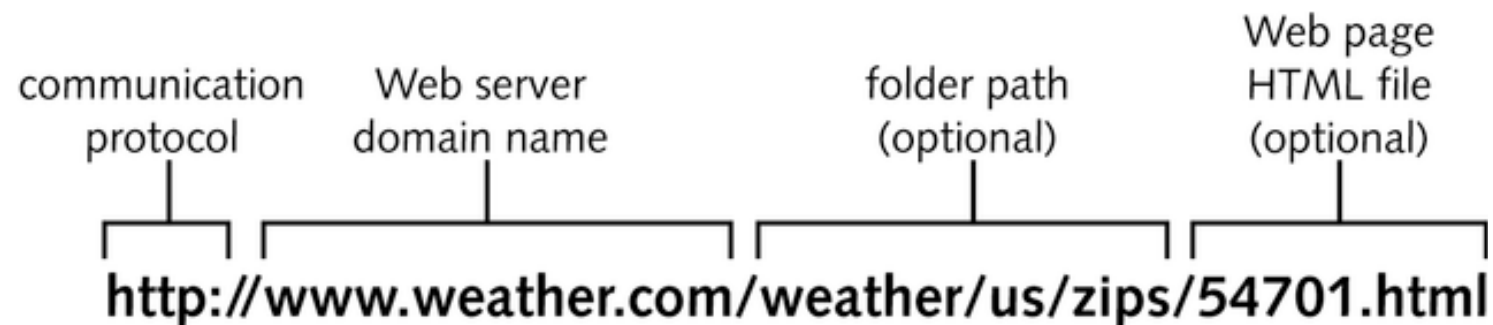


- Conventionally, client sends an HTTP request
- Server responds with an HTTP response
- AJAX & HTML5 has merged these roles somewhat

# HTTP URLs

---

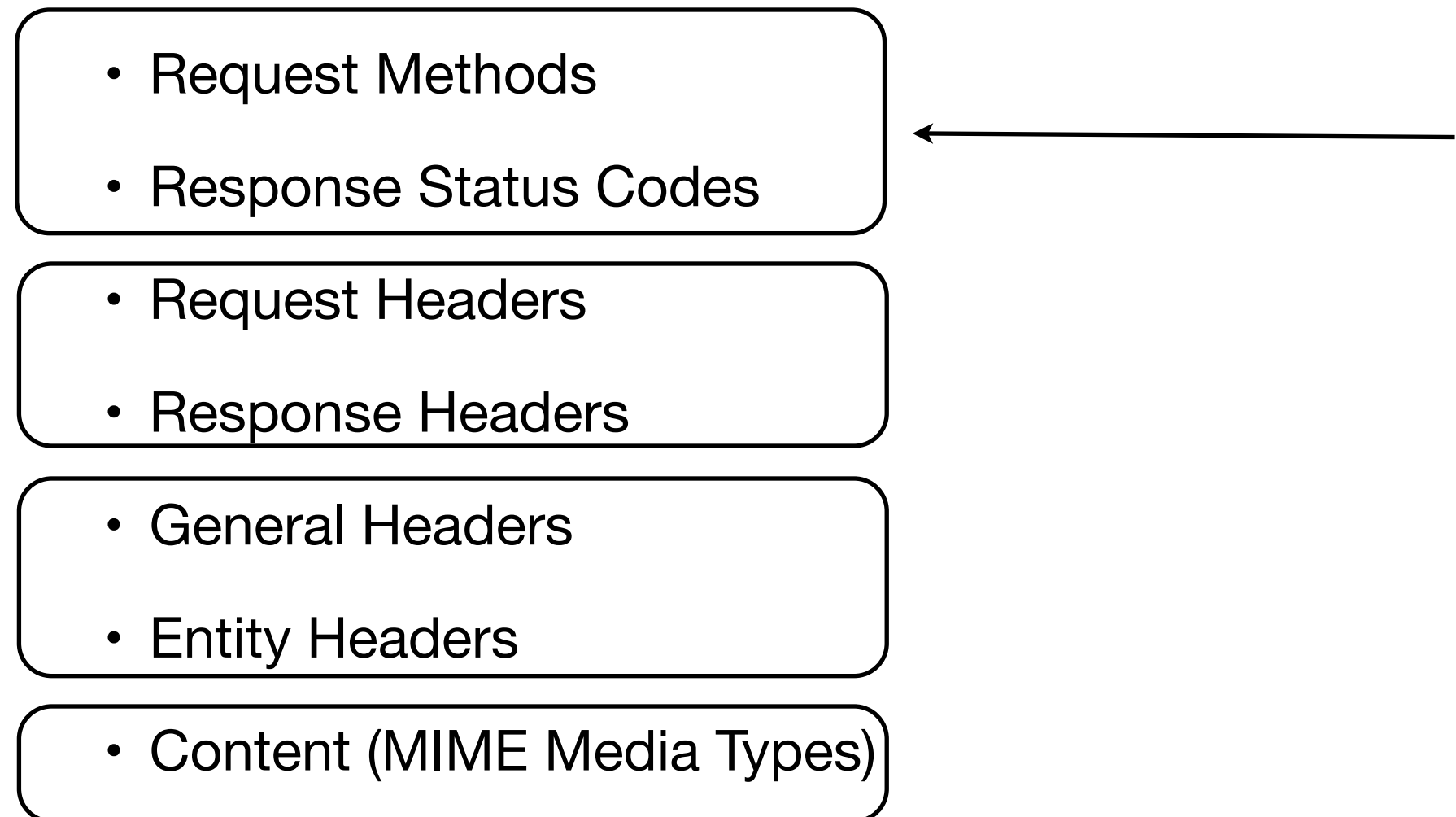
- HTTP is a request/response standard of a client and a server.
- Typically, an HTTP client initiates a request.
- Resources to be accessed by HTTP are identified using Uniform Resource Locaters (URLs).
- They contain four distinct parts: the protocol type, the machine name, the directory path and the file name.
- There are several kinds of URLs: file URLs, FTP URLs, and HTTP URLs.



# HTTP Components

---

- HTTP Requests and Response are comprised of these components



# HTTP Request Methods

- HTTP defines eight methods (sometimes referred to as "verbs") indicating the desired action to be performed on the identified resource.
- Two types :
  - Safe Methods -no side effects
    - HEAD: Request for data on resource.
    - GET: Request for resource.
    - TRACE: Echoes back the request just as it was received on the receiver for debugging.
    - OPTIONS: Used to check server capacity
  - Unsafe - intended for actions that have side effects
    - POST: Request for resource by passing parameters
    - PUT: Creation or sending of resource
    - DELETE: Deletion of resource
    - CONNECT: Reserved for use on intermediate servers that can operate as tunnels.

# Response Status Codes

---

- Indicate the server's disposition corresponding to a request
- Combination of a numerical code, and a short description
- Can be categorized in 5 categories:
  - 1xx--Informational
  - 2xx --Successful
  - 3xx--Redirection
  - 4xx--Client Error
  - 5xx--Server Error



## Request Headers

- Specific to an HTTP Request
- Carry information about the client, and the type of request
- Facilitates better understanding between client and server

Host  
Accept-Language  
If-Modified-Since  
Referer  
User-Agent  
Authorization  
If-None-Match  
Expect  
Accept  
Proxy-Authorization  
If-Range  
From  
Accept-Charset  
Max-Forwards  
If-Unmodified-Since  
TE  
Accept-Encoding  
If-Match  
Range

## Response Headers

- Specific to an HTTP Response
- Carry information about the server, and the type of response

Accept-Ranges  
ETag  
Retry-After  
WWW-Authenticate  
Age  
Location  
Server  
Proxy-Authenticate  
Vary

## General Headers

- Carry information about the HTTP transaction
- Can be a part of request, as well as response

Cache-Control  
Keep-Alive  
Pragma  
Via  
Connection  
Upgrade  
Trailer  
Warning  
Transfer-Encoding  
Date

## Entity Headers

- Carry information about the content
- Mainly a part of HTTP response

Allow  
Content-Language  
Content-Location  
Content-Range  
Content-Encoding  
Content-Length  
Content-MD5  
Content-Type  
Expires  
Last-Modified

---

# Content

- IANA maintains a list of valid content types
- It is specified by the Content-TypeEntity header
- Categorized in 9 MIME Media types

application  
audio  
example  
image  
message  
model  
multipart  
text

# Statelessness

---

- Because of the Connect, Request, Response, Disconnect nature of HTTP it is said to be a stateless protocol
- i.e. from one web page to the next there is nothing in the protocol that allows a web program to maintain program “state” (like a desktop program).
- “state” can be maintained by various techniques
- E.g: Cookies:
  - are text files stored by client browser
  - maintain session by storing information
  - are non-executable

# GET Example

---

```
GET /index.html HTTP/1.1
Host: www.example.com
User-Agent: Mozilla/4.5 [en]
Accept: image/gif, image/jpeg, text/html
Accept-language: en
Accept-Charset: iso-8859-1
```

# Request Line

---

- Request line: contains the requested resource. Consists of

1. Method: name of HTTP method called (GET, POST, etc.).
2. Resource identifier: URL (Uniform Resource Locator) of the requested resource.
3. Protocol version: protocol version requested for the response.

```
GET /index.html HTTP/1.1
```

```
Host: www.example.com
```

```
User-Agent: Mozilla/4.5 [en]
```

```
Accept: image/gif, image/jpeg, text/html
```

```
Accept-language: en
```

```
Accept-Charset: iso-8859-1
```

# Request Header

---

- Contains additional information to help the server to process the request correctly.
- Examples:
  - Host: name of requested server.
  - User-Agent: name of browser or program used to access the resource.
  - Accept: some text and image formats accepted by the client.
  - Accept-Language: languages supported (preferred) by the client, useful for automatically personalising the response.

```
GET /index.html HTTP/1.1
Host: www.example.com
User-Agent: Mozilla/4.5 [en]
Accept: image/gif, image/jpeg, text/html
Accept-language: en
Accept-Charset: iso-8859-1
```

# Request Parameters (1)

---

- A HTTP request can also contain parameters. e.g:
  - as a response to a registration form, the selection of a product in an online store, etc.
- These parameters can be passed in two ways:
  - As part of the request chain encrypted as part of the URL
  - As extra request data



# Request Parameters: as part of request chain

---

- To encrypt parameters as part of the URL, they are added to the URL after the name of the resource, separated from the latter by the character ?.
  - The different parameters are separated from one another by the character &.
  - Spaces are replaced by +.
  - Special characters are represented by %xx where xx represents the hexadecimal ASCII code of the character

url

```
http://www.example.com/index.php?name=Mr+Nobody&OK=1
```

request

```
GET (/index.php?name=Mr+Nobody&OK=1) HTTP/1.0
Host: www.example.com
User-Agent: Mozilla/4.5 [en]
Accept: image/gif, image/jpeg, text/html
Accept-language: en
Accept-Charset: iso-8859-1
```

# Extra Request Data

---

- To pass the parameters as the body of the request, the POST method rather than GET needs to be used.

```
POST /index.php HTTP/1.0
Host: www.example.com
User-Agent: Mozilla/4.5 [en]
Accept: image/gif, image/jpeg, text/html
Accept-language: en
Accept-Charset: iso-8859-1
```

```
name=Mr+Nobody&OK=1
```

# Response

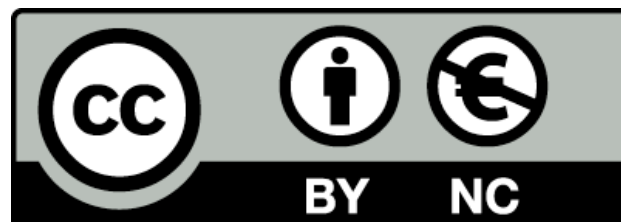
---

- Response Code

```
HTTP/1.1 200 OK
Date: Mon, 04 Aug 2011 15:19:10 GMT
Server: Apache/2.0.40 (Red Hat Linux)
Last-Modified: Tue, 25 Mar 2011 08:52:53 GMT
Accept-Ranges: bytes
Content-Length: 428
```

- HTML

```
<html>
<body>
<h1>My Dvd List</h1>
(more file contents)
.
.
.
</body>
</html>
```



Except where otherwise noted, this content is licensed under a Creative Commons Attribution-NonCommercial 3.0 License.

For more information, please see <http://creativecommons.org/licenses/by-nc/3.0/>



Waterford Institute of Technology  
INSTITIÚID TEICNEOLAÍOCHTA PHORT LÁIRGE

