# HTTP: the hypertext transfer protocol

Presenter: Assist.Prof.Dr. Fatih ABUT

# HTTP vs HTML

- HTML: hypertext markup language
  - Definitions of tags that are added to Web documents to control their appearance
- HTTP: hypertext transfer protocol
  - The rules governing the conversation between a Web client and a Web server

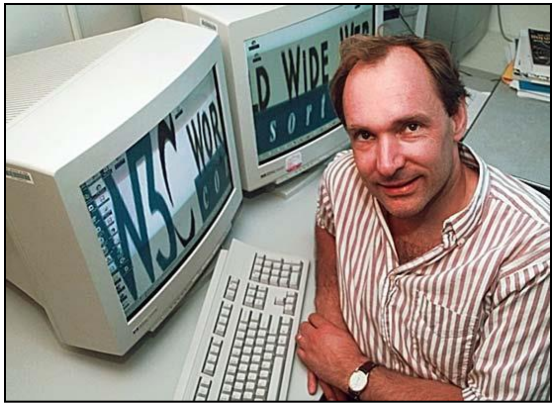
Both were invented at the same time by the same person

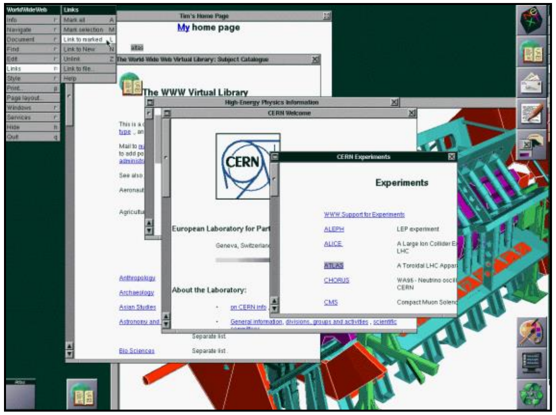
# HTTP History

Design Criteria:
- Simplicity
- Efficiency
- Performance
- Port consumption

- 1990 – World Wide Web project
  - Tim Berners-Lee starts project at CERN
  - Demonstrates browser/editor accessing hypertext files
  - HTTP 0.9 defined, supports only hypertext, linked to port 80 (no images)
- 1991 – first web server outside Europe
  - CERN releases WWW, installed at SLAC
- 1992 – HTTP 1.0, supports images, scripts as well
- 1993 – Growth phase
- 1994 – CERN and MIT agree to set up WWW Consortium
- 1999 – HTTP 1.1, supports open ended extensions

# Tim Berners-Lee





**My** home page

The WWW Virtual Library

This is a
type , an

Mail to
to add po
administr

See also

Aeronaut

Agricultu

Anthropology

Archaeology

Asian Studies

Astronomy and

Separate list

Bio Sciences Separate list

European Laboratory for Part

Geneva, Switzerlan

About the Laboratory:

- on CERN info
- General information, divisions, groups and activities , scientific

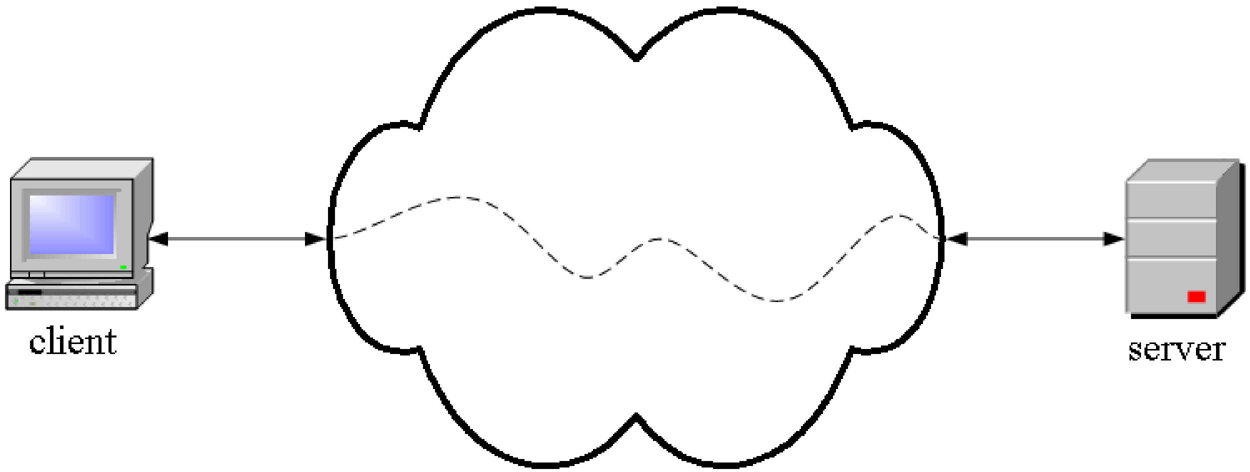
Experiments

WWW Support for Experiments

| | |
|---|---|
| ALEPH | LEP experiment |
| ALICE | A Large Ion Collider E<br>LHC |
| ATLAS | A Toroidal LHC Appar |
| CHORUS | WA95 - Neutrino oscil<br>CERN |
| CMS | Compact Muon Solen |

Tim Berners-Lee was knighted by Queen Elizabeth for his invention of the World Wide Web. He is shown here, along with the first picture posted on the Web and a screen shot from an early version of his Web browser.

# HTTP is an application layer protocol



- The Web client and the Web server are application programs
- Application layer programs do useful work like retrieving Web pages, sending and receiving email or transferring files
- Lower layers take care of the communication details
- The client and server send messages and data without knowing anything about the communication network

# The application layer is boss – the top layer

| **Layer** | **Function** |
|---|---|
| Application | Do useful work like Web browsing, email, and file transfer |
| Lower layers | Handle communication between the client and server |

- Your boss says: Send this package to Miami -- I don't care if you use Federal Express, UPS, or any other means. Also, let me know when it arrives or if it cannot be delivered for some reason.

- The application program says: Send this request to the server -- I don't care how you do it or whether it goes over phone lines, radio, or anything else about the details. Just send the message, and let me know when it arrives or if it cannot be delivered for some reason.

# Many application layer protocols are used on the Internet, HTTP is only one

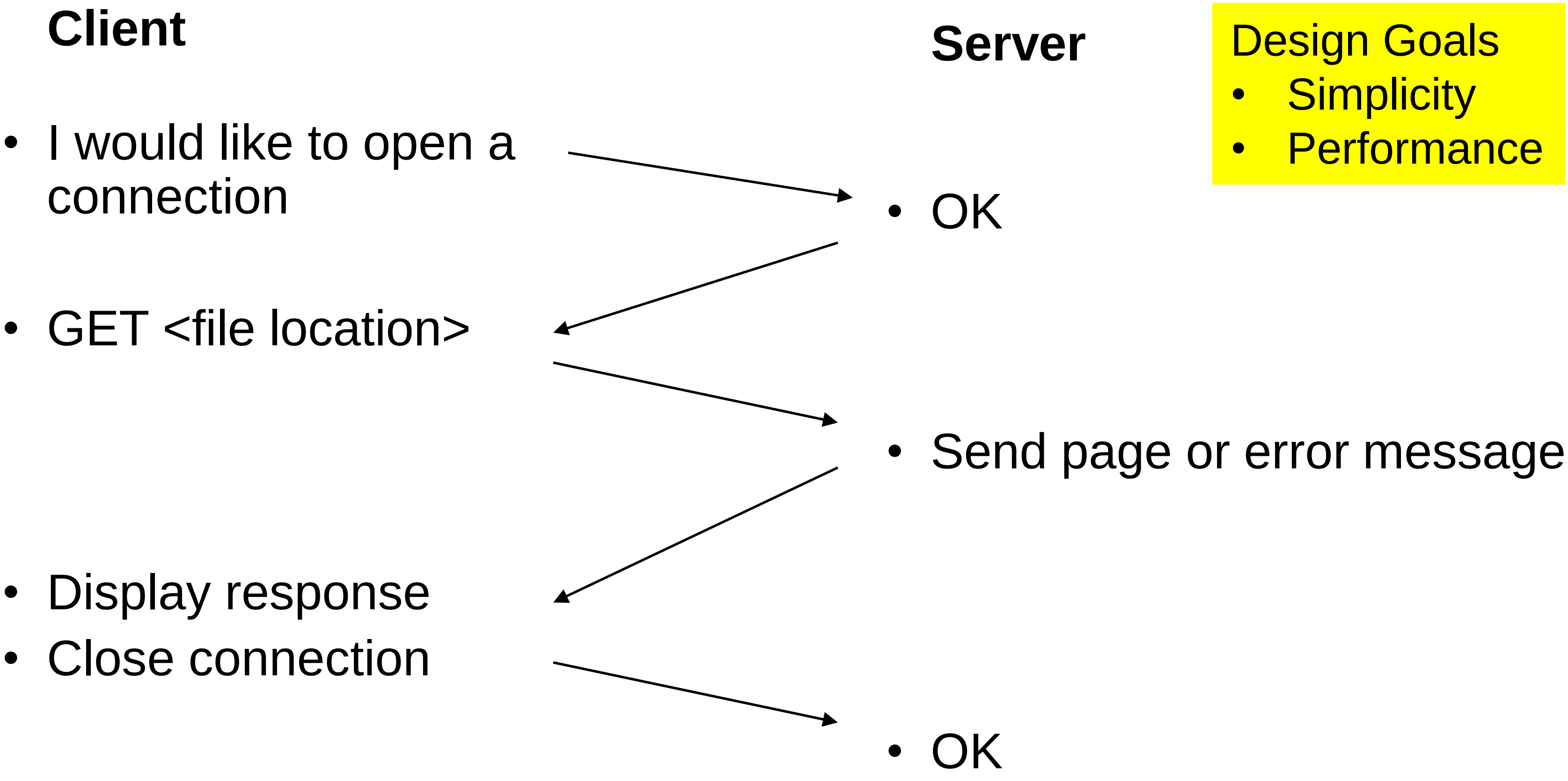
| **Protocol** | **Application** |
|---|---|
| HTTP: Hypertext Transfer | Retrieve and view Web pages |
| FTP: File Transfer | Copy files from client to server or from server to client |
| SMTP: Simple Mail Transport | Send email |
| POP: Post Office | Read email |

# The TCP/IP protocol layers

The application program is king – it gets work done using the lower level layers for communication between the client and server.

| | |
|---|---|
| Application | Get useful work done – retrieve Web pages, copy files, send and receive email, etc. |
| Transport | Make client-server connections and optionally control transmission speed, check for errors, etc. |
| Internet | Route packets between networks |
| Data link | Route data packets within the local area network |
| Physical | Specify what medium connects two nodes, how binary ones and zeros are differentiated, etc, |

# An HTTP conversation



**Client**

- I would like to open a connection
- GET <file location>
- Display response
- Close connection

**Server**

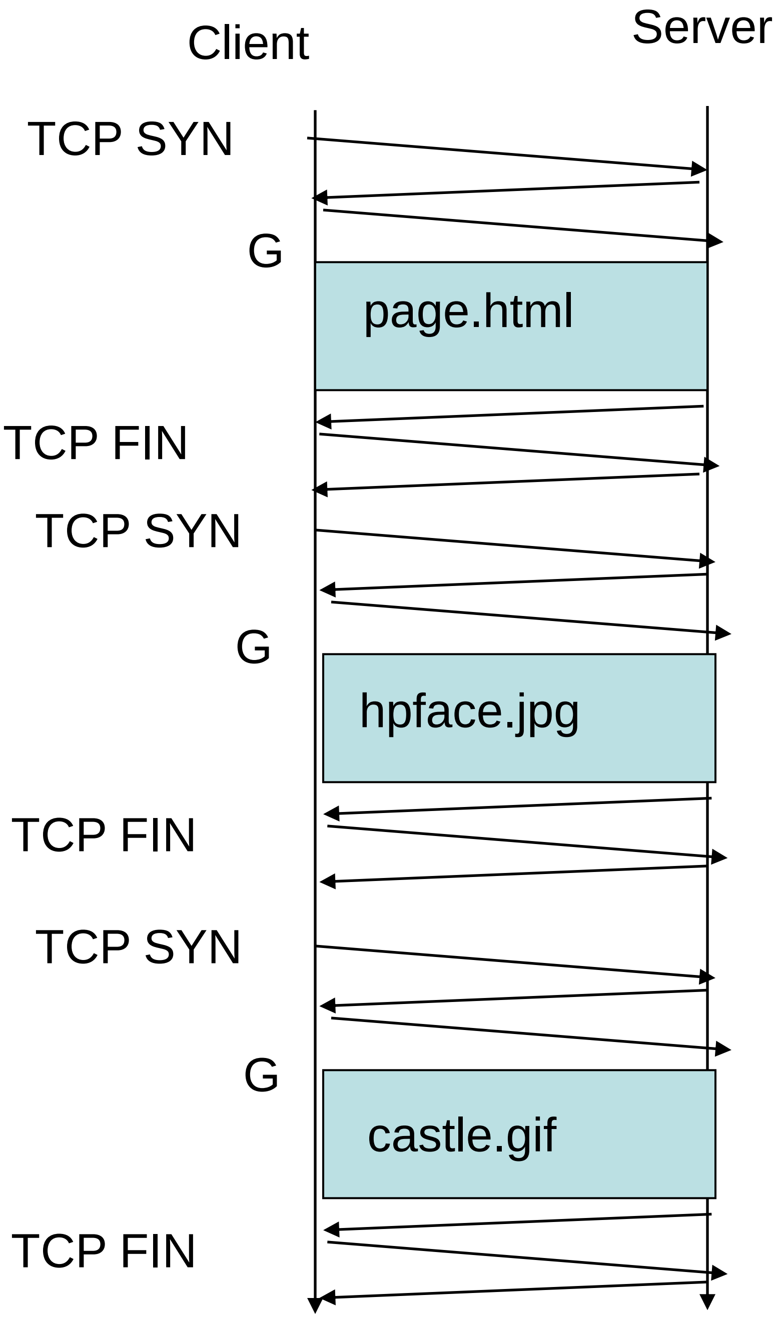
- OK
- Send page or error message
- OK

Design Goals
- Simplicity
- Performance

HTTP is the set of rules governing the format and content of the conversation between a Web client and server

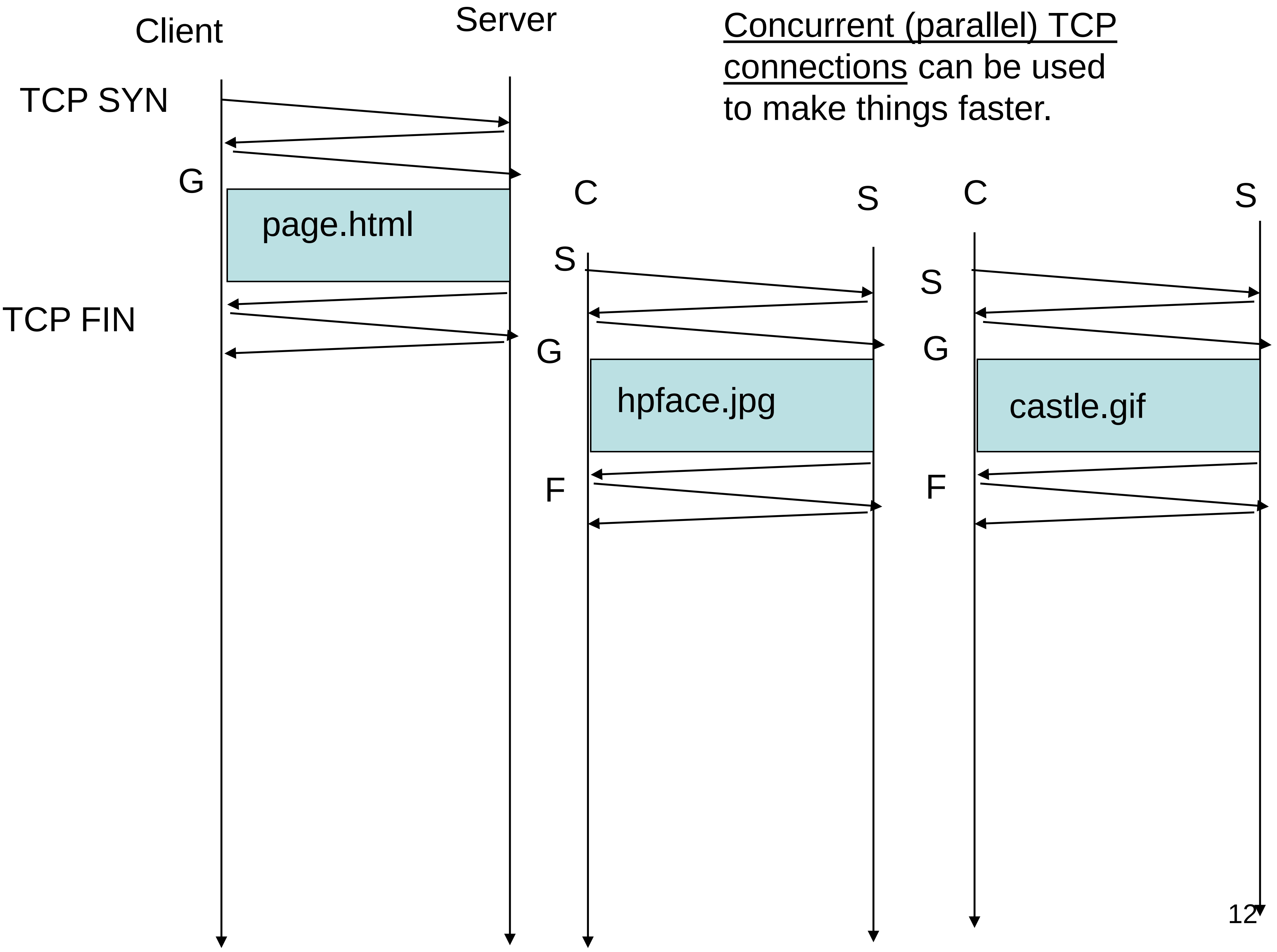
# Possible Communication Models for HTTP

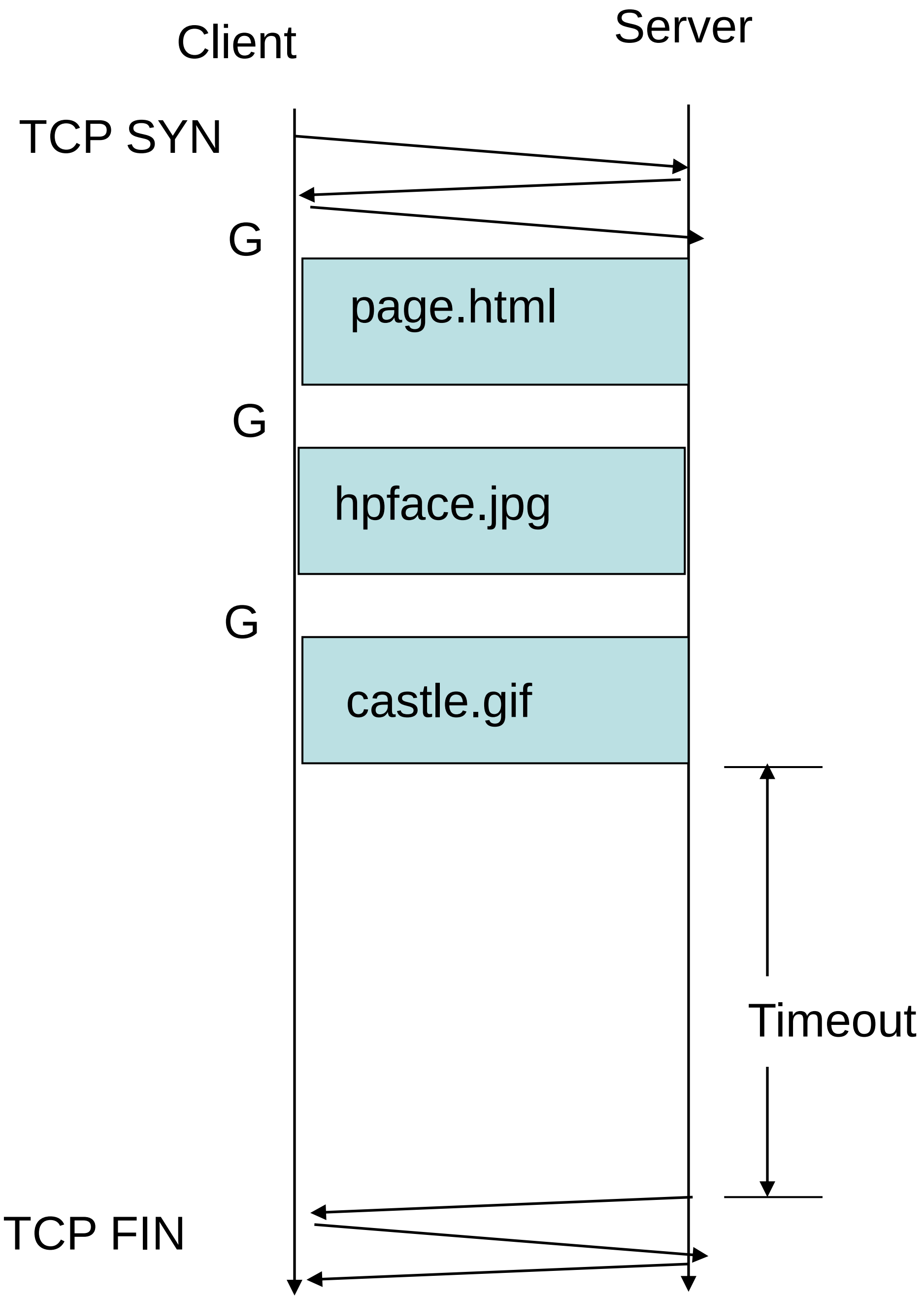
- The “classic” approach: HTTP/1.0
- HTTP over multiple, concurrent TCP connections
- Persistent-HTTP: HTTP/1.1
- HTTP Pipelining
- Transaction-TCP (T/TCP)
- HTTP over UDP-based protocols



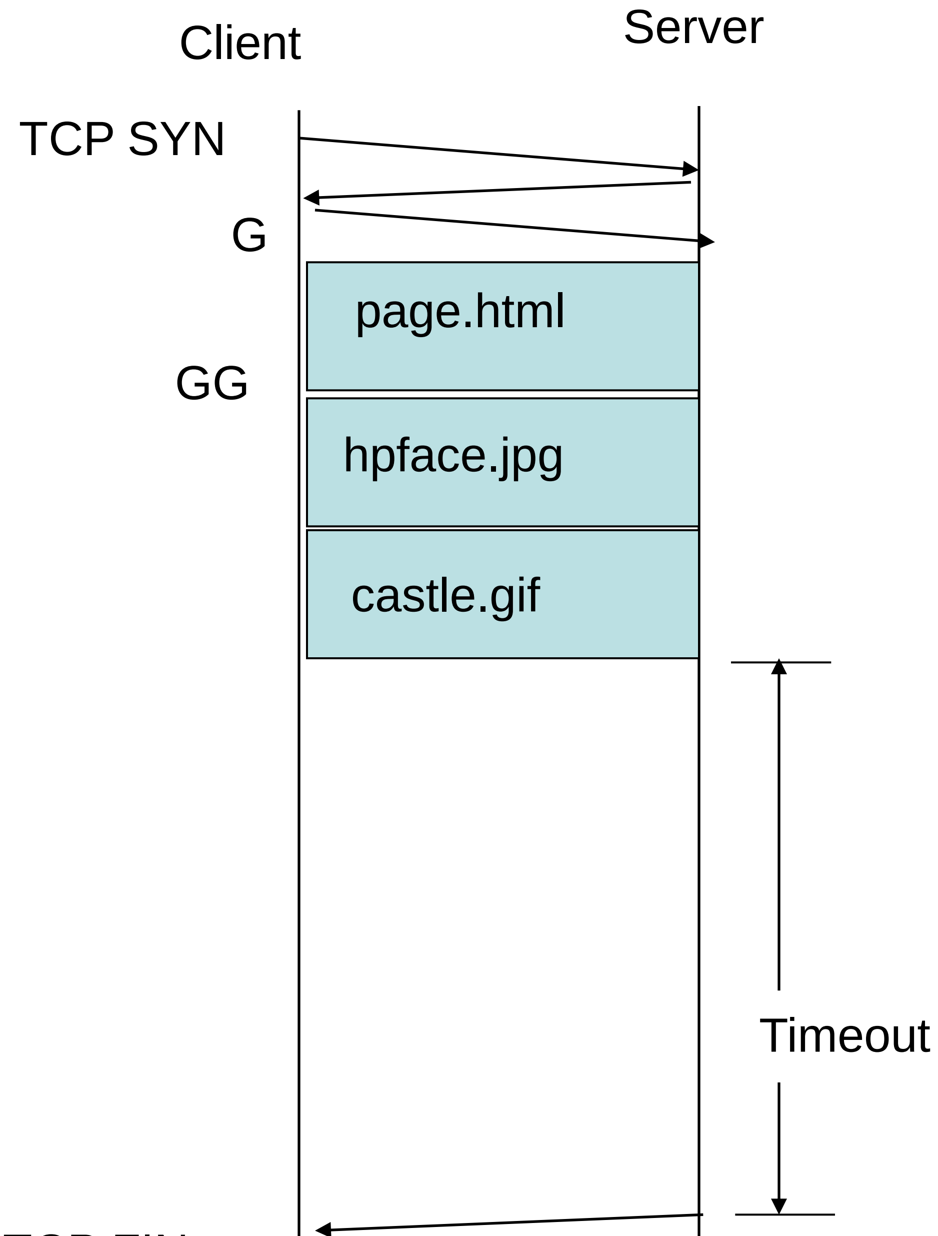
The “classic” approach in HTTP/1.0 is to use one HTTP request per TCP connection, serially.

Concurrent (parallel) TCP connections can be used to make things faster.



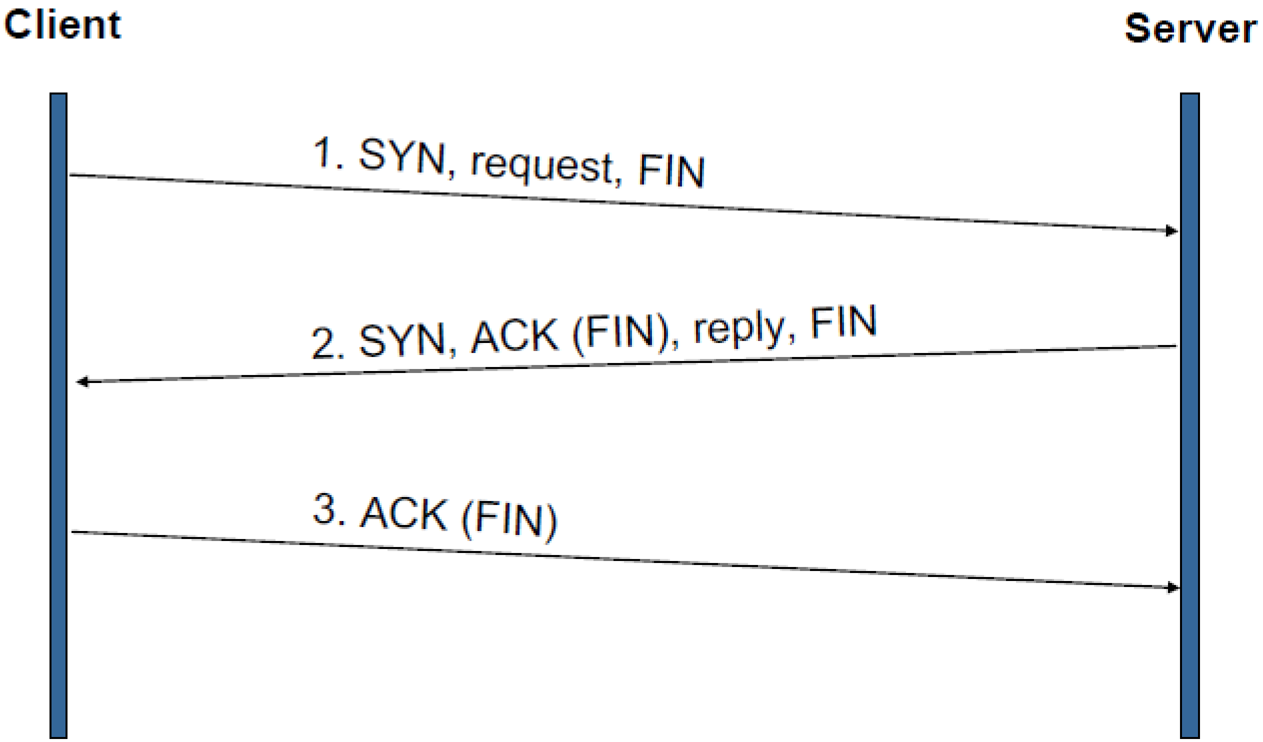


The “persistent HTTP” approach can re-use the same TCP connection for Multiple HTTP transfers, one after another, serially. Amortizes TCP overhead, but maintains TCP state longer at server.



The “pipelining” feature in HTTP/1.1 allows requests to be issued asynchronously on a persistent connection. Requests must be processed in proper order. Can do clever packaging.

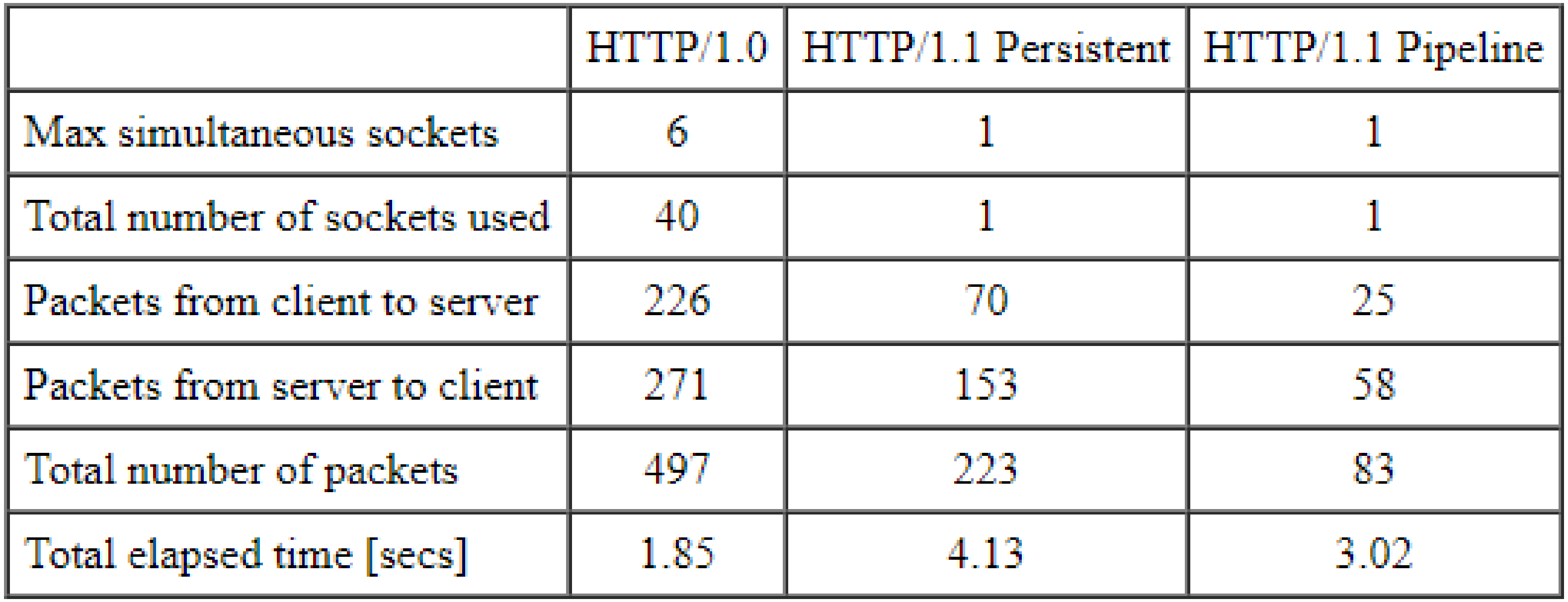
# Transaction-TCP



# HTTP over UDP

- What is Motivation?
- What are the Advantages?
- What are the Disadvantages?
- What are the Consequences?

# Performance Effects of HTTP/1.1



| | HTTP/1.0 | HTTP/1.1 Persistent | HTTP/1.1 Pipeline |
|---|---|---|---|
| Max simultaneous sockets | 6 | 1 | 1 |
| Total number of sockets used | 40 | 1 | 1 |
| Packets from client to server | 226 | 70 | 25 |
| Packets from server to client | 271 | 153 | 58 |
| Total number of packets | 497 | 223 | 83 |
| Total elapsed time [secs] | 1.85 | 4.13 | 3.02 |

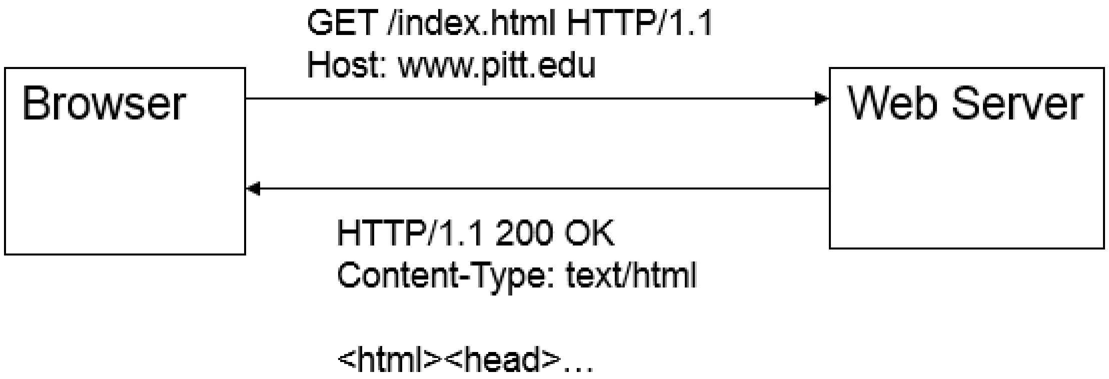
**Nielsen, Hendrik Frystyk, Gettys, Jim, Baird-Smith, Anselm, Prud'hommeaux, Eric, Håkon Wium Lie, Lilley, Chris: Network Performance Effects of HTTP/1.1, CSS1, and PNG, W3C, NOTE 24-June 1997.**

# Properties of HTTP 1.1

- Based on TCP
- Persistent connections
- Request-/Response Protocol
- Bidirectional Data Flow
- Stateless
- Not Session-Oriented

# HTTP Messages (1)

- HEAD
- **GET**
- **POST**
- PUT
- DELETE
- TRACE
- OPTIONS
- CONNECT

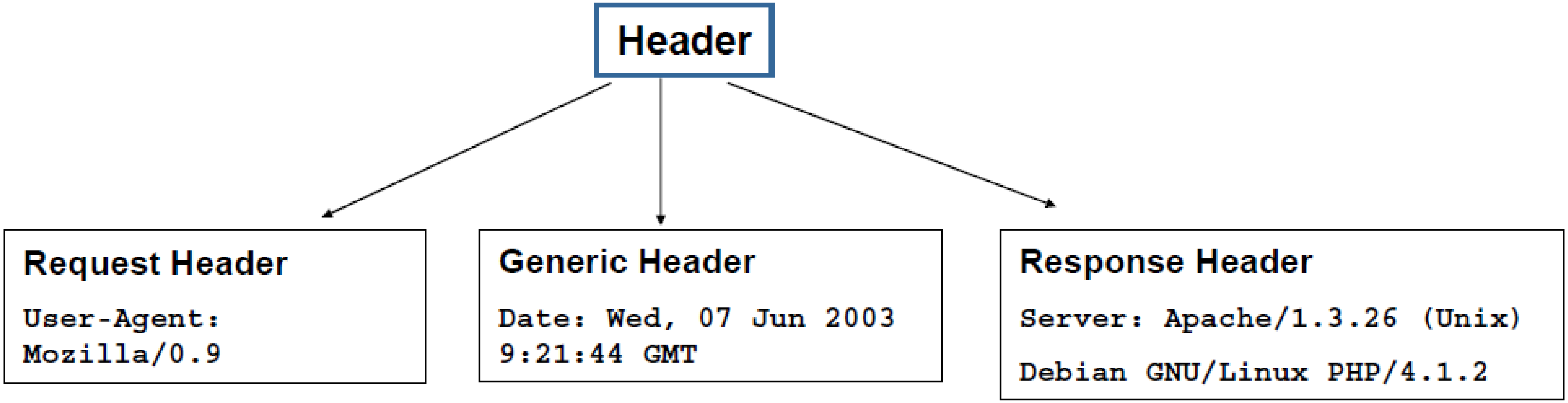


```
GET /index.html HTTP/1.1
Host: www.pitt.edu
```

```
HTTP/1.1 200 OK
Content-Type: text/html

<html><head>…
```

# HTTP Messages (2)



Header

Request Header

```
User-Agent:
Mozilla/0.9
```

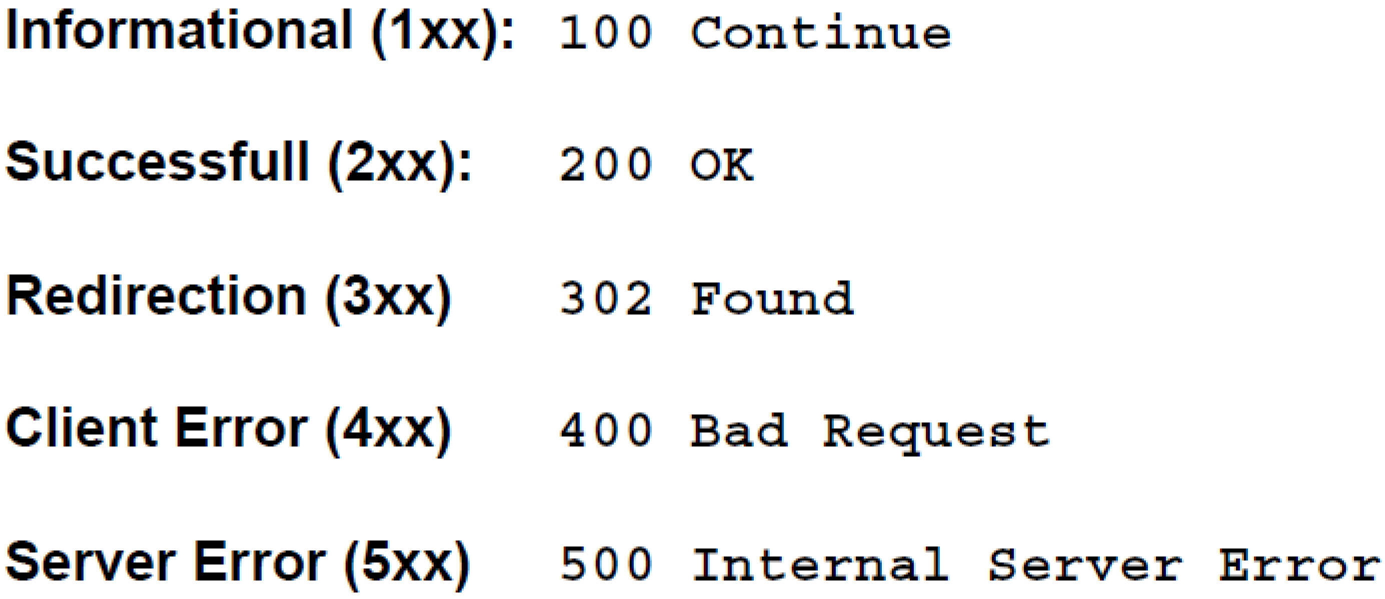
Generic Header

```
Date: Wed, 07 Jun 2003
9:21:44 GMT
```

Response Header

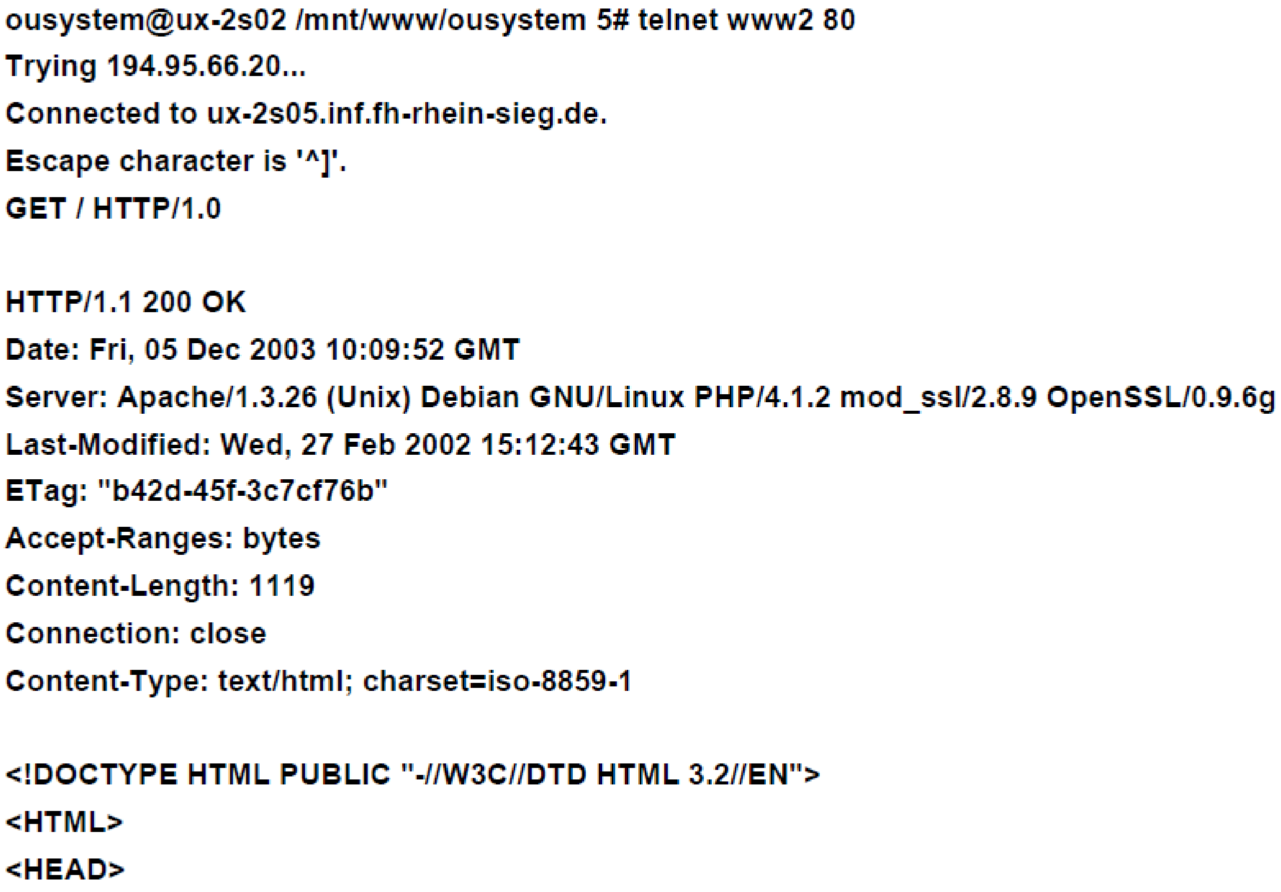
```
Server: Apache/1.3.26 (Unix)
Debian GNU/Linux PHP/4.1.2
```

# HTTP Messages (3)



| | |
|---|---|
| **Informational (1xx):** | `100 Continue` |
| **Successfull (2xx):** | `200 OK` |
| **Redirection (3xx)** | `302 Found` |
| **Client Error (4xx)** | `400 Bad Request` |
| **Server Error (5xx)** | `500 Internal Server Error` |

# HTTP Messages (4)



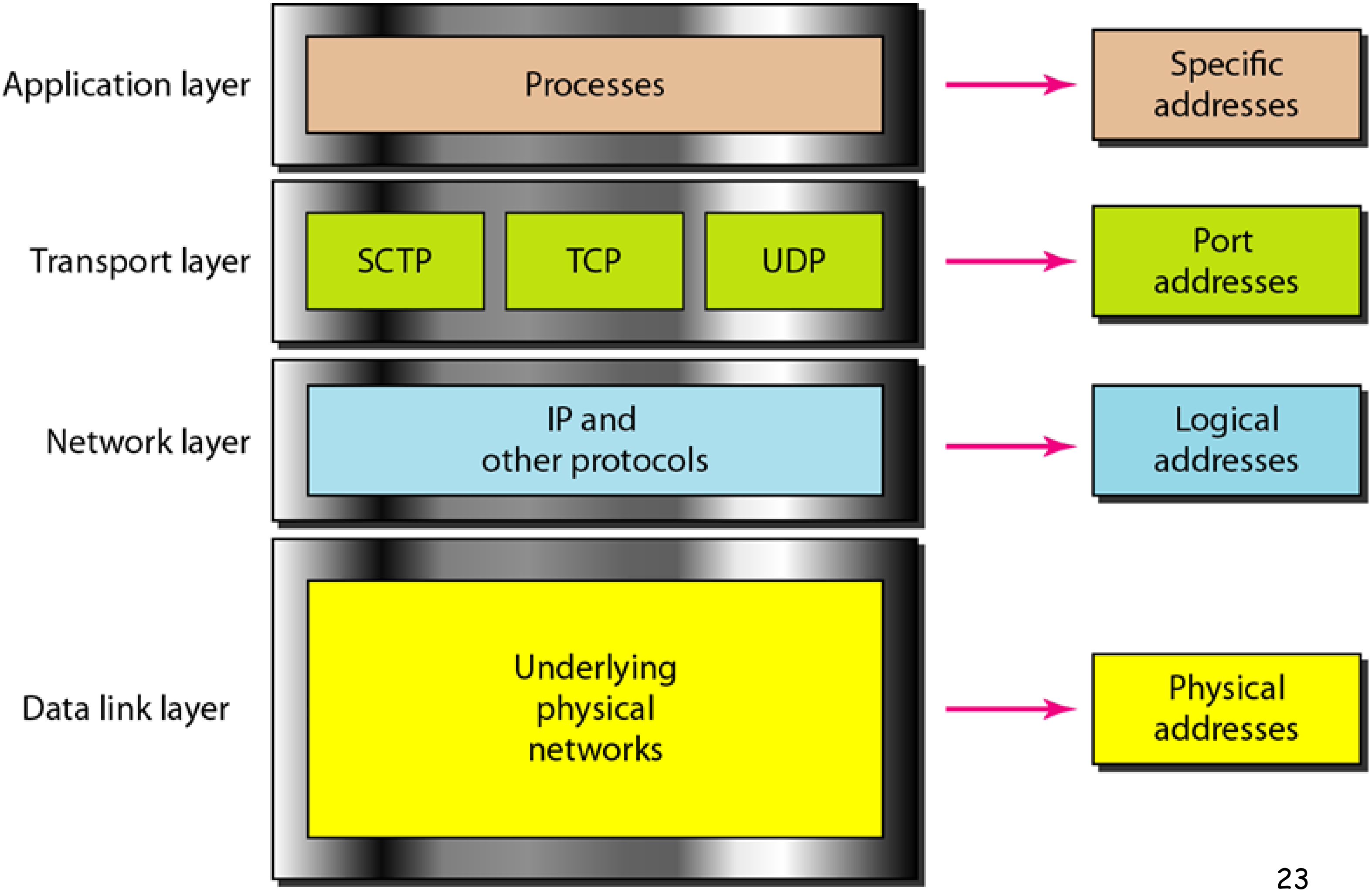
```
ousystem@ux-2s02 /mnt/www/ousystem 5# telnet www2 80
Trying 194.95.66.20...
Connected to ux-2s05.inf.fh-rhein-sieg.de.
Escape character is '^]'.
GET / HTTP/1.0

HTTP/1.1 200 OK
Date: Fri, 05 Dec 2003 10:09:52 GMT
Server: Apache/1.3.26 (Unix) Debian GNU/Linux PHP/4.1.2 mod_ssl/2.8.9 OpenSSL/0.9.6g
Last-Modified: Wed, 27 Feb 2002 15:12:43 GMT
ETag: "b42d-45f-3c7cf76b"
Accept-Ranges: bytes
Content-Length: 1119
Connection: close
Content-Type: text/html; charset=iso-8859-1

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2//EN">
<HTML>
<HEAD>
```

# Relationship of layers and addresses in TCP/IP

HTTP Addressing?



# URI History

- Web Addresses [Berners-Lee] 1990
  - Universal Document Identifiers 1992
  - URI: Universal Resource Identifiers RFC 1630
  - URL: Locators RFC 1736,1738,1808
  - URN: Names RFC 1737,2141
  - URC: Characteristics RFC 1737

- A simple and extensible means of identification
  - http://www.ics.uci.edu:80/~fielding/talks/
  - mailto:fielding@ics.uci.edu

# URI Principles

- Establishing identity by reference
  - name, moniker, location, …
  - global scope
- Simple
  - Just a string of common characters
- Transcribable
  - a sequence of characters, not coded character octets
- Uniformity
  - Identifying various resources using same scheme