

IT ARCHITECTURE

WEB ARCHITECTURE

Willy Sudiarto Raharjo

Teknik Informatika
Universitas Kristen Duta Wacana

Table of Contents

- Internet
- World Wide Web
- HTTP
- Web Server
- Web Browser
- Web Architecture
- Web Application Architecture

Internet



Web Based Architecture

Lecture 123 Servers



- Presenter & Player Program Delivery
- Content Repository
- QA Collaboration
- Search Delivery

Communities Sharing Content and Collaborating

Content Producers



- Present
- Record
- Publish

Internet

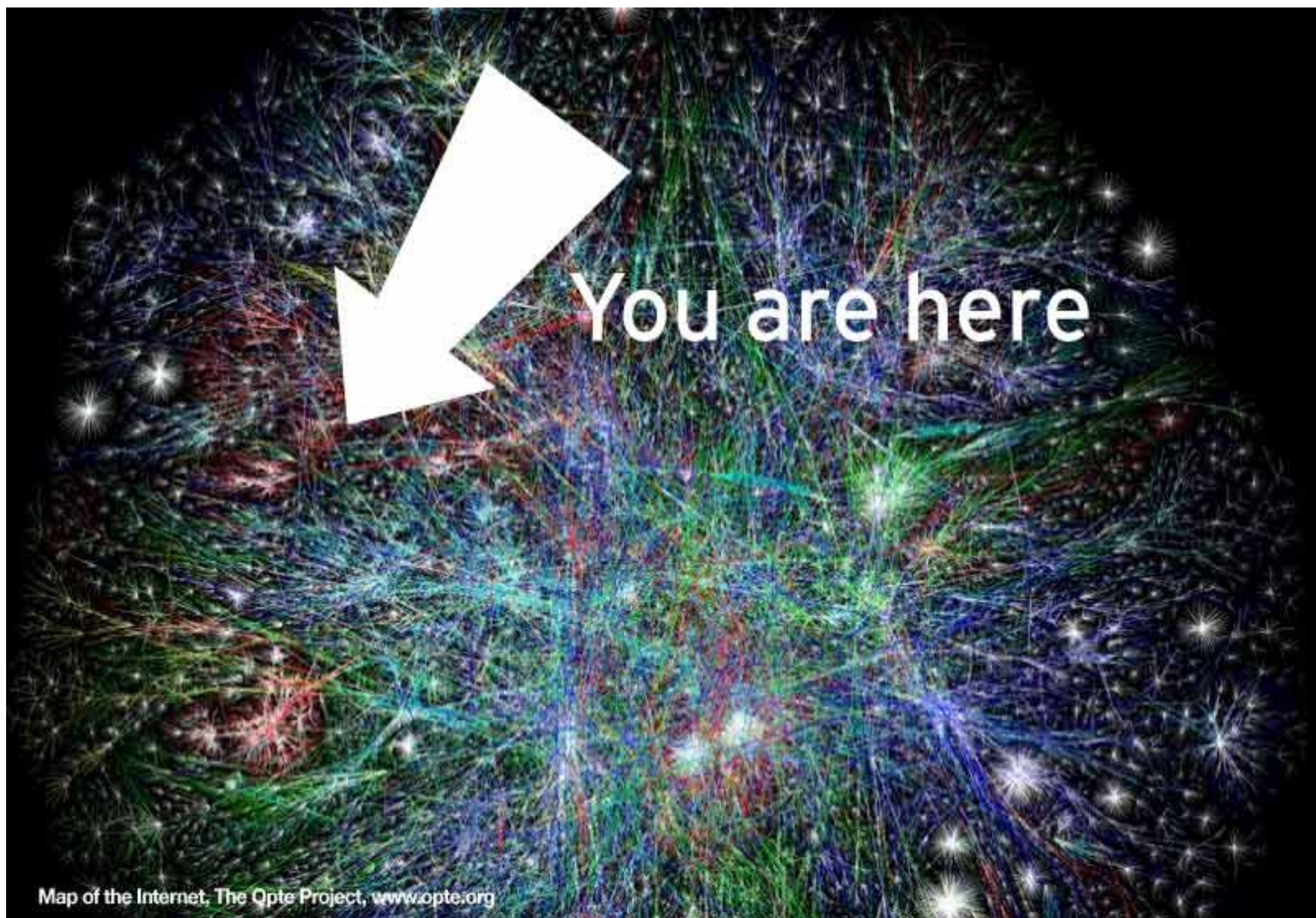


Viewers

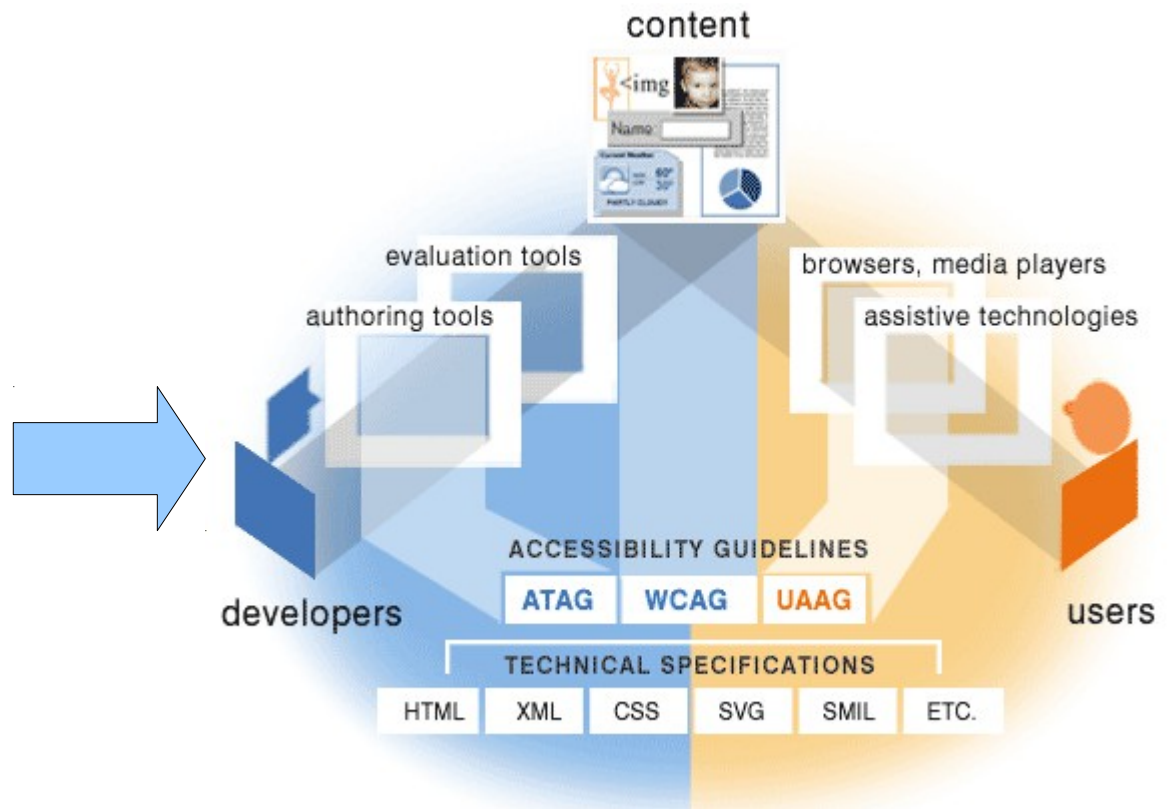


Playback

- Any computer
 - Any time
 - Any place
- Ask Questions



Tim Berners-Lee



First Web Page

World Wide Web

The WorldWideWeb (W3) is a wide-area [hypermedia](#) information retrieval initiative aiming to give universal access to a large universe of documents.

Everything there is online about W3 is linked directly or indirectly to this document, including an [executive summary](#) of the project, [Mailing lists](#) , [Policy](#) , November's [W3 news](#) , [Frequently Asked Questions](#) .

[What's out there?](#)

Pointers to the world's online information, [subjects](#) , [W3 servers](#), etc.

[Help](#)

on the browser you are using

[Software Products](#)

A list of W3 project components and their current state. (e.g. [Line Mode](#) ,X11 [Viola](#) , [NeXTStep](#) , [Servers](#) , [Tools](#) , [Mail robot](#) , [Library](#))

[Technical](#)

Details of protocols, formats, program internals etc

[Bibliography](#)

Paper documentation on W3 and references.

[People](#)

A list of some people involved in the project.

[History](#)

A summary of the history of the project.

[How can I help ?](#)

If you would like to support the web..

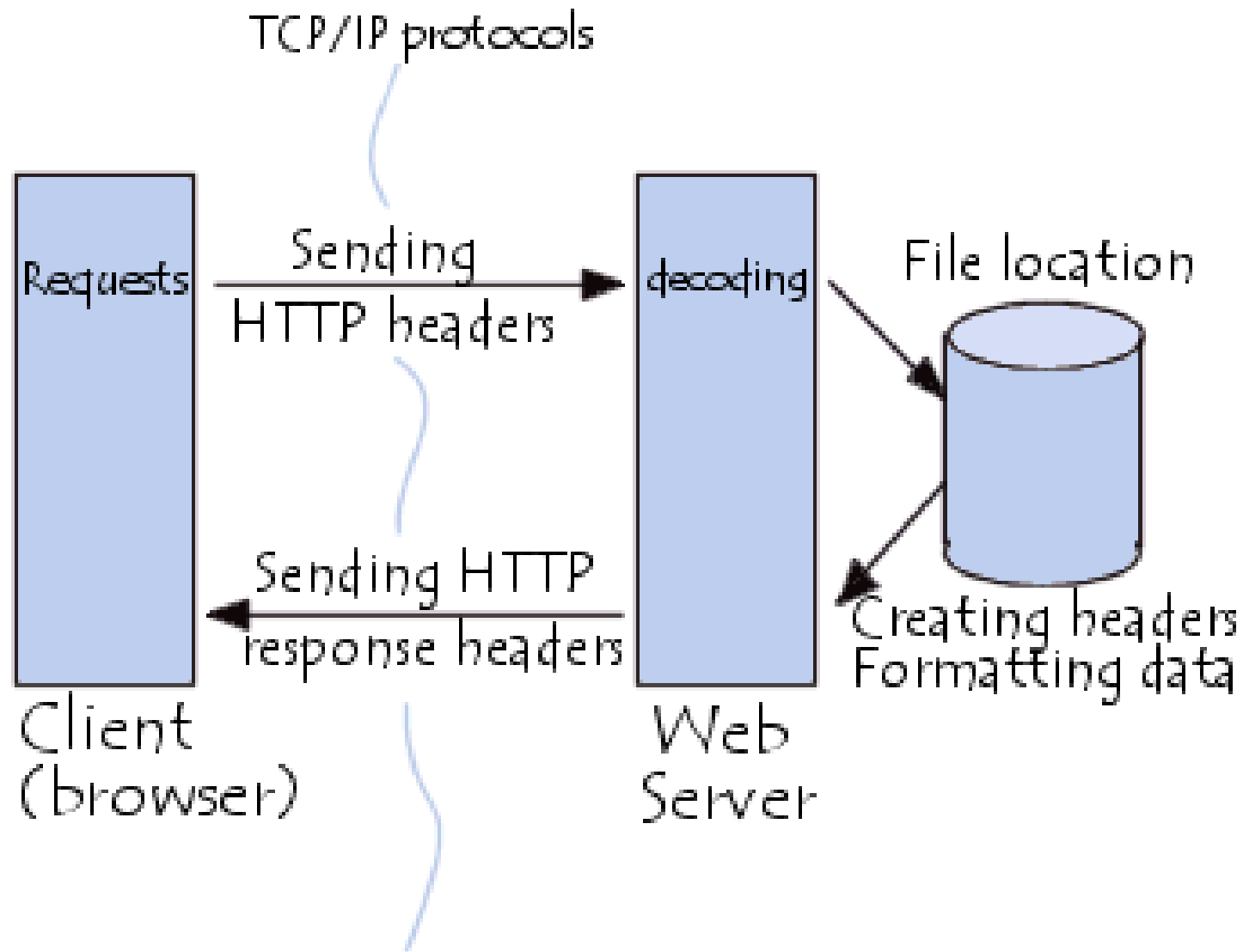
[Getting code](#)

Getting the code by [anonymous FTP](#) , etc.

Current Web Pages



HTTP Protocol



HTTP is Stateless

FIGURE 6-9

HTTP messages flow between a browser and a Web server.

1. The URL in the browser's Address bar contains the domain name of the Web server that your browser contacts.

Address `www.infoweblinks.com/np/chapter6.html`

2. Your browser opens a socket and connects to a similar open socket at the Web server.

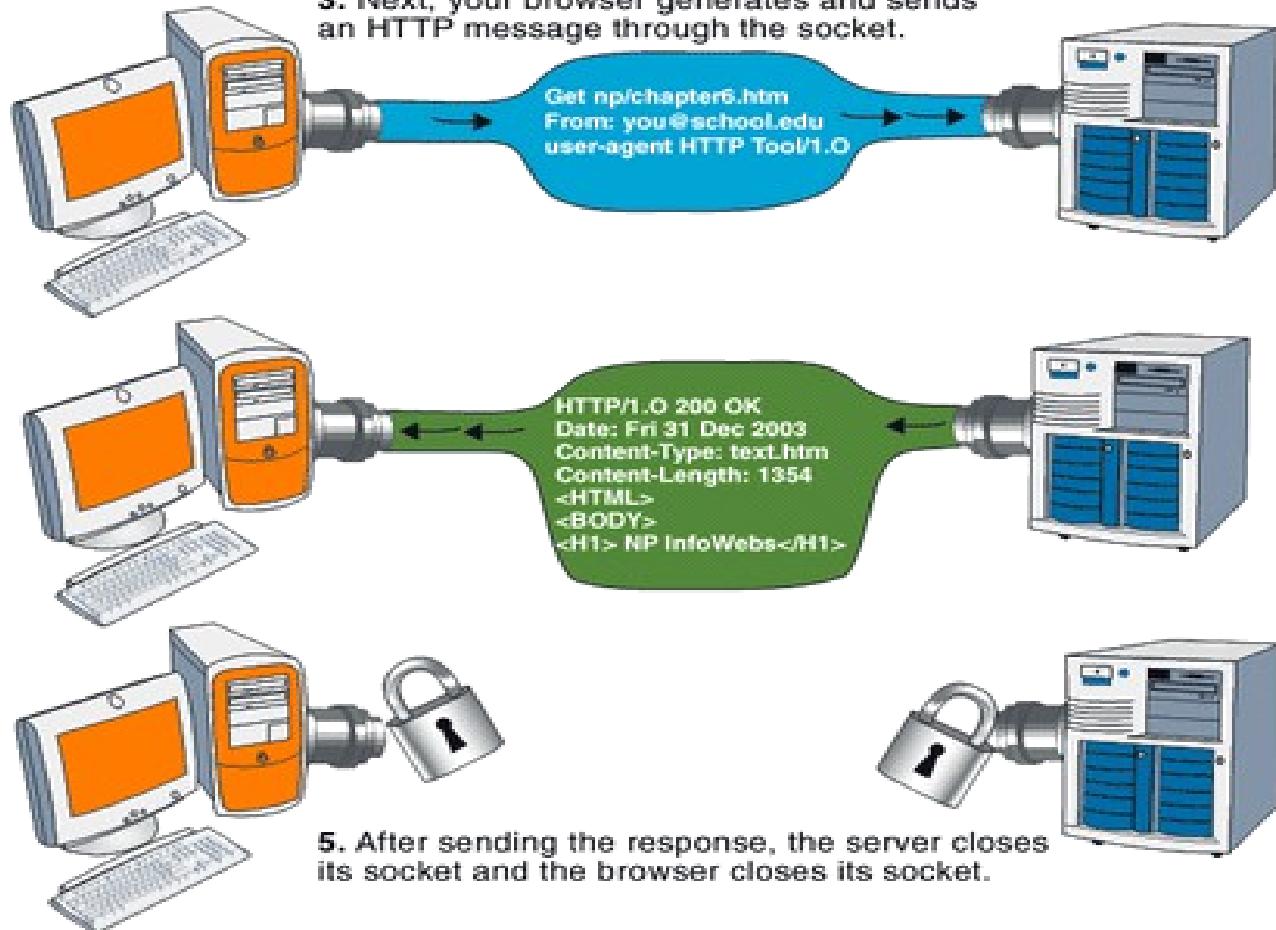
3. Next, your browser generates and sends an HTTP message through the socket.

Get np/chapter6.htm
From: you@school.edu
user-agent HTTP Tool/1.0

4. The server sends back the requested HTML document through the open sockets.

HTTP/1.0 200 OK
Date: Fri 31 Dec 2003
Content-Type: text/html
Content-Length: 1354
<HTML>
<BODY>
<H1> NP InfoWebs</H1>

5. After sending the response, the server closes its socket and the browser closes its socket.



Solution: Cookies



New in HTTP 1.1

Proxy support and the Host field:

HTTP 1.1 has a required Host header by spec.

HTTP 1.0 does not officially require a Host header, but it doesn't hurt to add one, and many applications (proxies) expect to see the Host header regardless of the protocol version.

Example:

```
GET / HTTP/1.1
Host: www.blahblahblahblah.com
```

This header is useful because it allows you to route a message through proxy servers, and also because your a web server can distinguish between different sites on the same server.

So this means if you have blahblahbah.com and helohelohelo.com both pointing to the same IP. Your web server can use the Host field to distinguish which site the client machine wants.

Persistent connections:

HTTP 1.1 also allows you to have persistent connections which means that you can have more than one request/response on the same HTTP connection.

In HTTP 1.0 you had to open a new connection for each request/response pair. And after each response the connection would be closed. This lead to some big efficiency problems because of [TCP Slow Start](#).

OPTIONS method:

HTTP/1.1 introduces the OPTIONS method. An HTTP client can use this method to determine the abilities of the HTTP server. Although it is not very much used today, most of this information is passed on server responses.

Caching:

HTTP 1.0 had support for caching via the header: If-Modified-Since.

HTTP 1.1 expands on the caching support a lot by using something called 'entity tag'. If 2 resources are the same, then they will have the same entity tags.

HTTP 1.1 also adds the If-Unmodified-Since, If-Match, If-None-Match conditional headers.

There are also further additions relating to caching like the Cache-Control header.

100 Continue status:

There is a new return code in HTTP/1.1 100 Continue. This is to prevent a client from sending a large request when that client is not even sure if the server can process the request, or is authorized to process the request. In this case the client sends only the headers, and the server will tell the client 100 Continue, go ahead with the body.

Much more:

- Digest authentication and proxy authentication
- Extra new status codes
- Chunked transfer encoding
- Connection header
- Enhanced compression support
- Much much more.

HTTP-message = Request | Response ; HTTP/1.1 messages

generic-message = start-line
 *(message-header CRLF)
 CRLF [message-body]

start-line = Request-Line | Status-Line

message-header = field-name ":" [field-value] *'host:' mandatory*

Request-Line = Method SP Request-URI SP HTTP-Version CRLF

Method = "OPTIONS" ; Section 9.2
 | "GET" ; Section 9.3
 | "HEAD" ; Section 9.4
 | "POST" ; Section 9.5
 | "PUT" ; Section 9.6
 | "DELETE" ; Section 9.7
 | "TRACE" ; Section 9.8
 | "CONNECT" ; Section 9.9
 | extension-method

HTTP Request Method

GET - Fetch a document

- Ex: URL variabel

POST - Execute the document, using the data in body document

- Ex: form submit

HEAD - Fetch just the header of the document

- Ex: date modified

PUT - Store a new document on the server

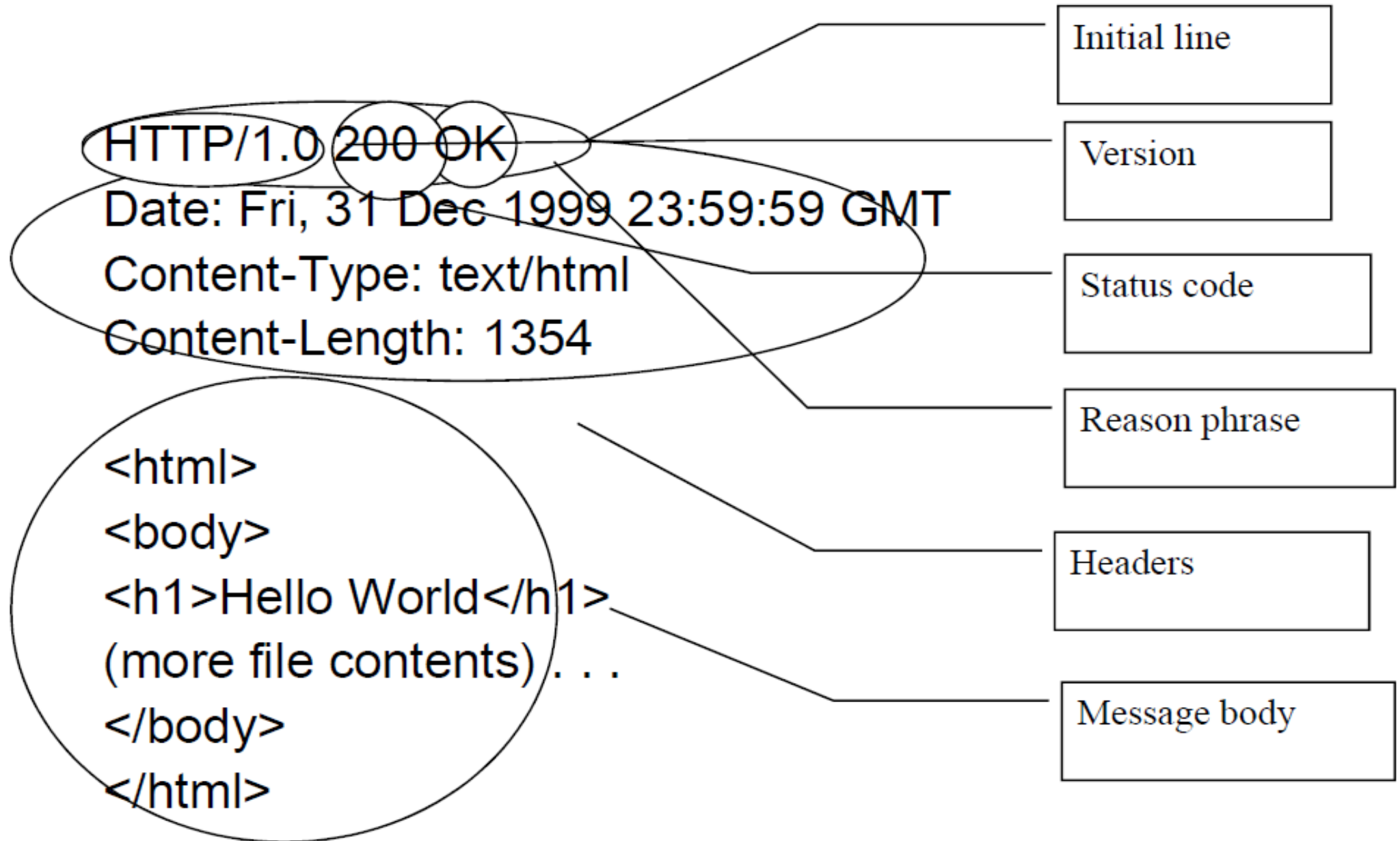
- Ex: upload using WebDAV

DELETE - Remove a document from the server

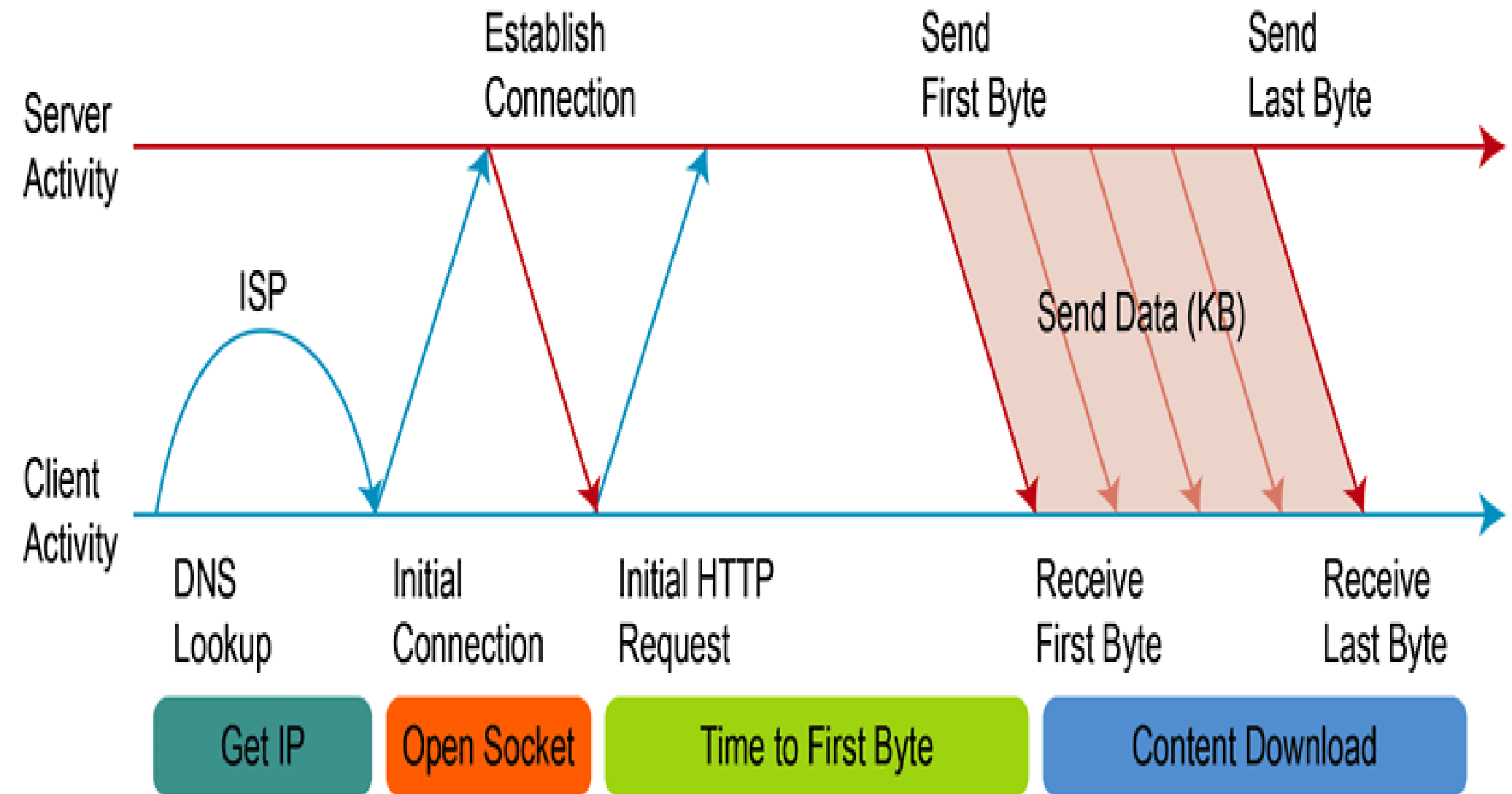
HTTP Response Code

Response Code	HTTP Operation	Response Body Contents	Description
200	GET, PUT, DELETE	Resource	No error, operation successful.
201 Created	POST	Resource that was created	Successful creation of a resource.
202 Accepted	POST, PUT, DELETE	N/A	The request was received.
204 No Content	GET, PUT, DELETE	N/A	The request was processed successfully, but no response body is needed.
301 Moved Permanently	GET	XHTML with link	Resource has moved.
303 See Other	GET	XHTML with link	Redirection.
304 Not Modified	conditional GET	N/A	Resource has not been modified.
400 Bad Request	GET, POST, PUT, DELETE	Error Message	Malformed syntax or a bad query.
401 Unauthorized	GET, POST, PUT, DELETE	Error Message	Action requires user authentication.
403 Forbidden	GET, POST, PUT, DELETE	Error Message	Authentication failure or invalid Application ID.
404 Not Found	GET, POST, PUT, DELETE	Error Message	Resource not found.
405 Not Allowed	GET, POST, PUT, DELETE	Error Message	Method not allowed on resource.

HTTP Response Example



The HTTP Request



HTTP Response

HTTP/1.1 200 OK

Status Line

Date: Thu, 20 May 2004 21:12:58 GMT

Connection: close

General Headers

Server: Apache/1.3.27

Accept-Ranges: bytes

Response Headers

Content-Type: text/html

Content-Length: 170

Last-Modified: Tue, 18 May 2004 10:14:49 GMT

Entity Headers

<html>

<head>

<title>Welcome to the Amazing Site!</title>

</head>

<body>

<p>This site is under construction. Please come back later. Sorry!</p>

</body>

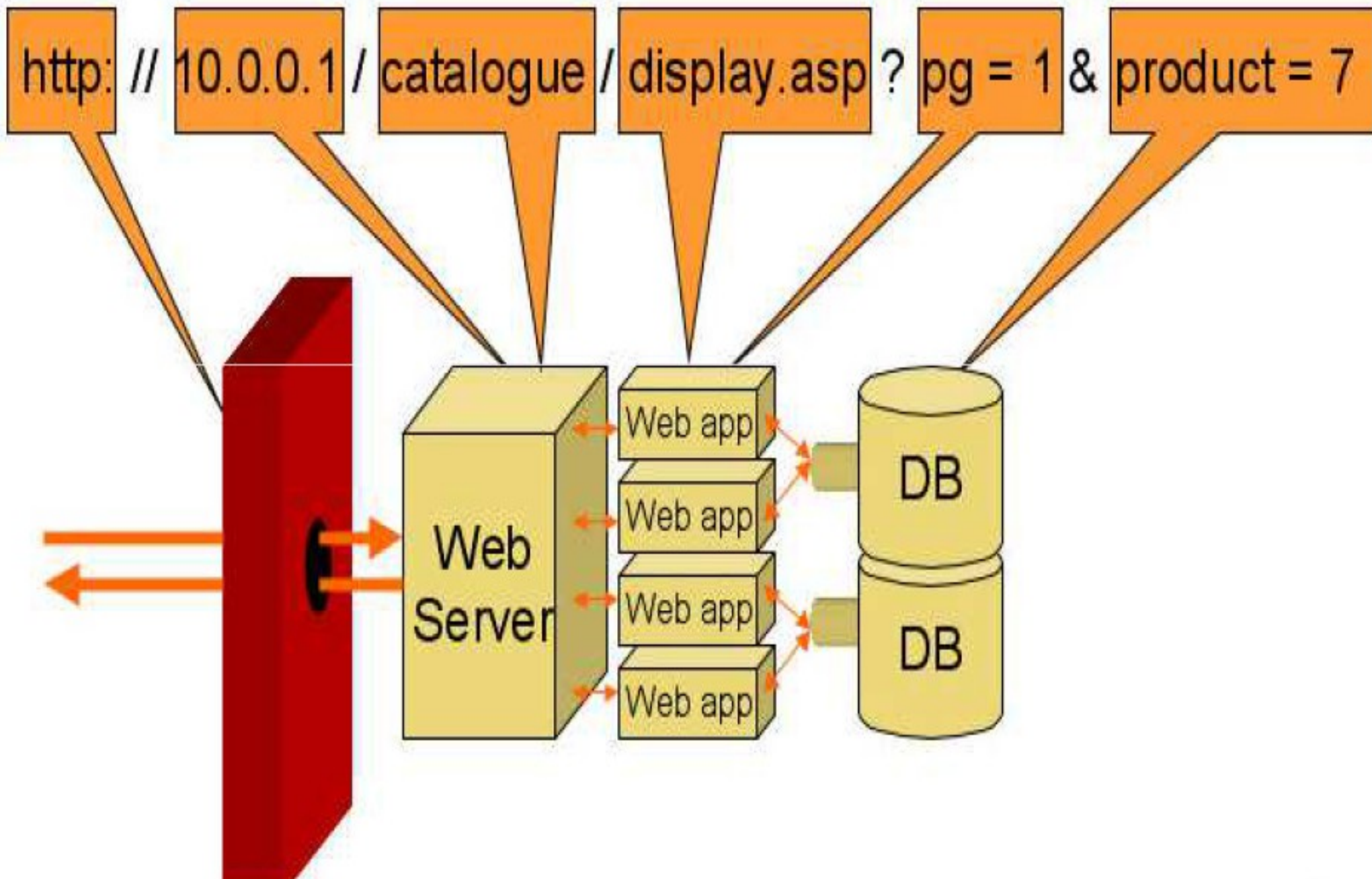
</html>

Message Body

**HTTP
Response**

Web Servers

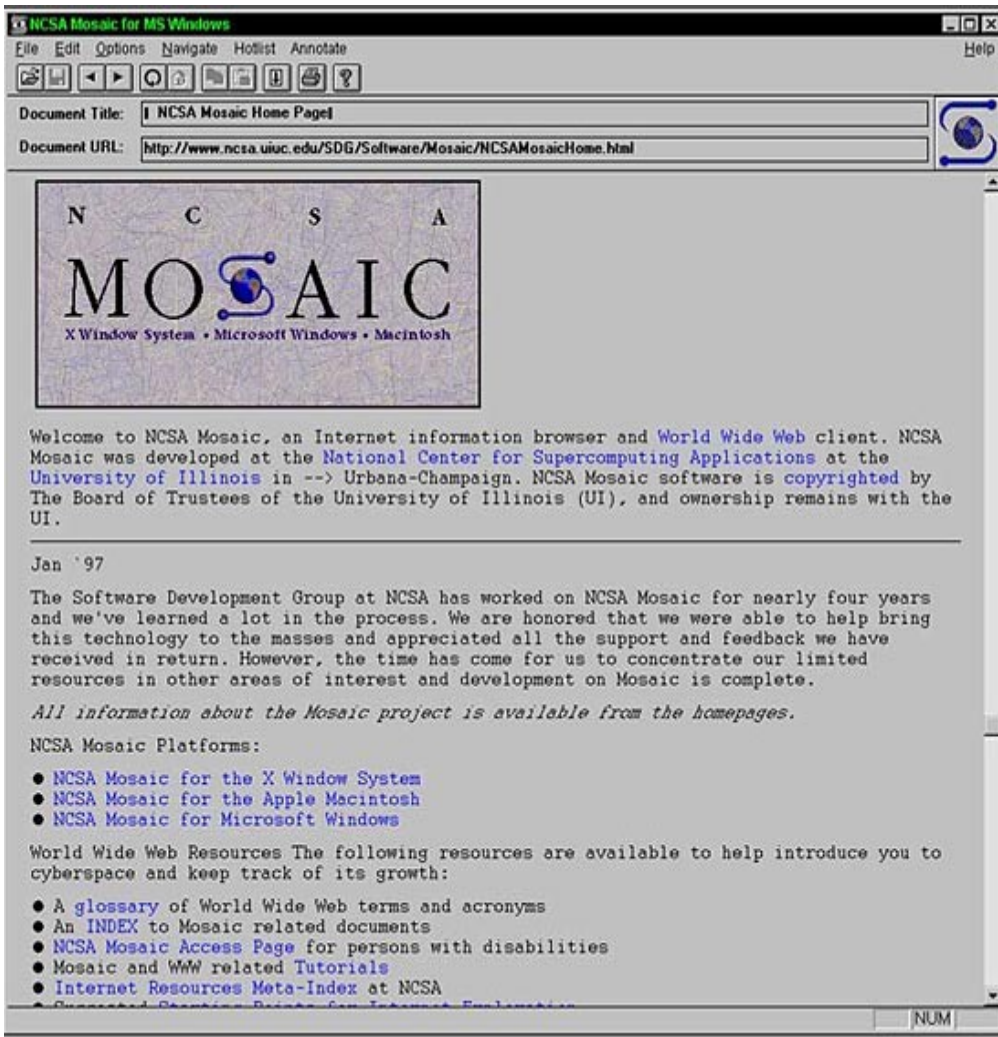
- Virtual hosting: multiple domain
- Address mapping: map request address to the actual file on server
- Authentication: by password, host credentials
- Handling MIME (multi purpose internet mail extensions)
- Caching support
- Security support



Web Server Hardware Architecture

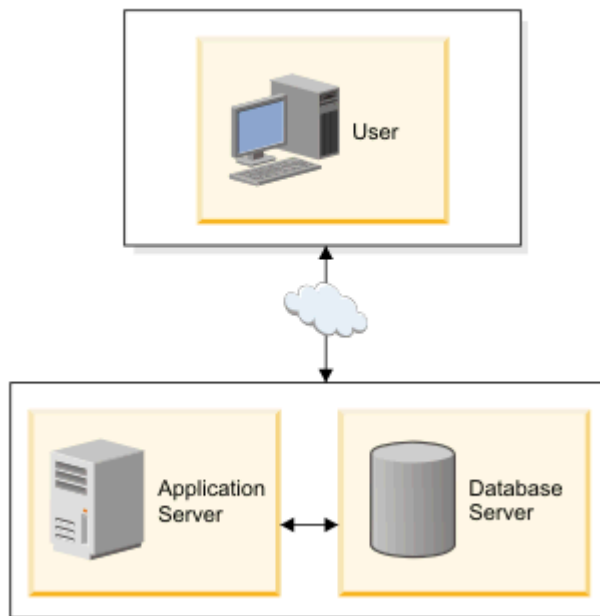
- **Server farms**
 - Large collections of servers
- **Centralized architecture**
 - Uses a few very large and fast computers
- **Distributed/decentralized architecture**
 - Uses a large number of less powerful computers
 - Divides the workload among them

Web Browsers

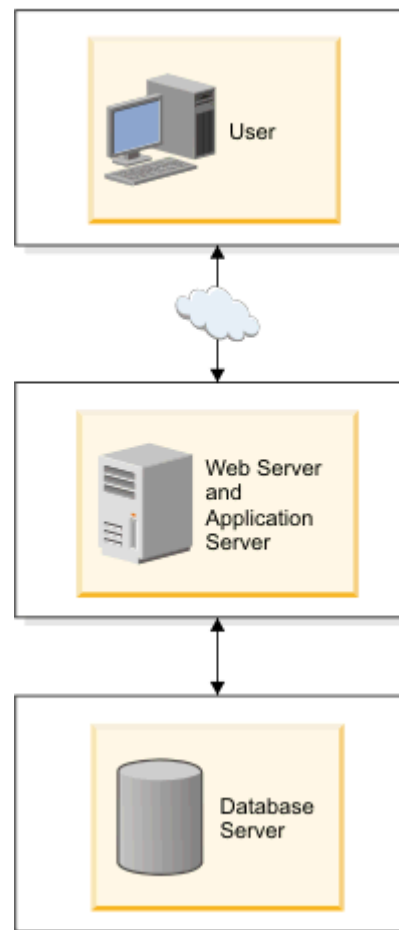


Web Architectures

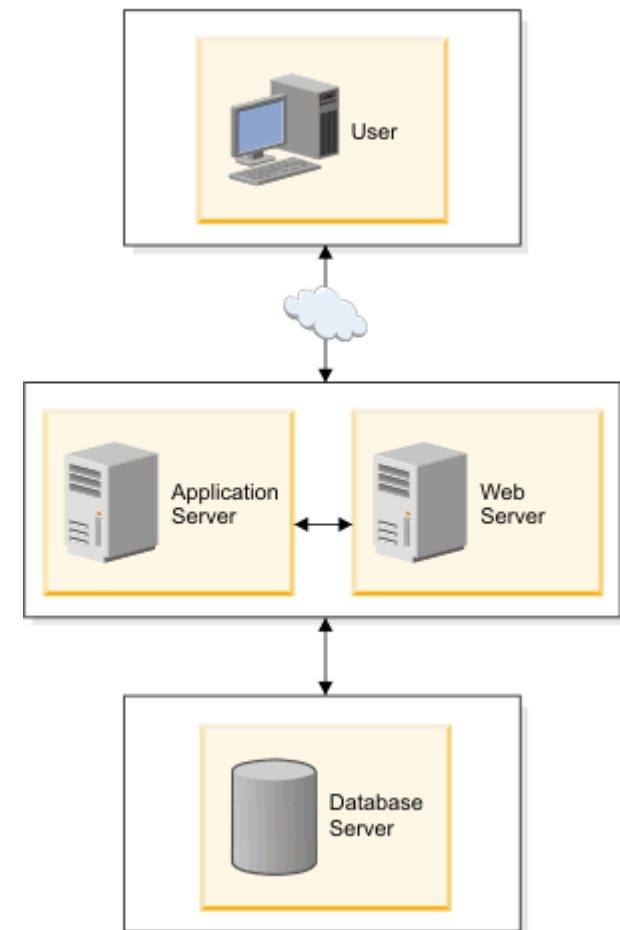
2 Tier



3 Tier A

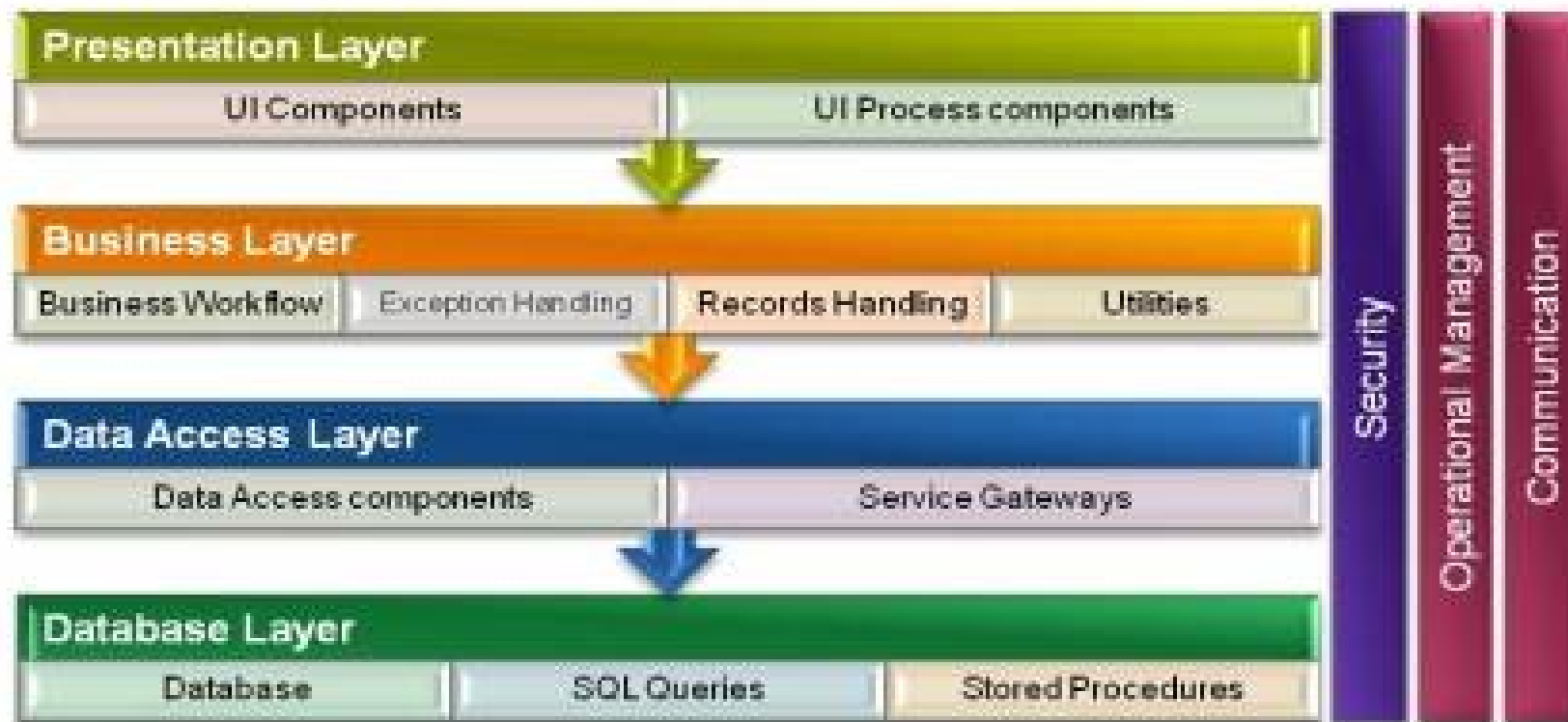


3 Tier B

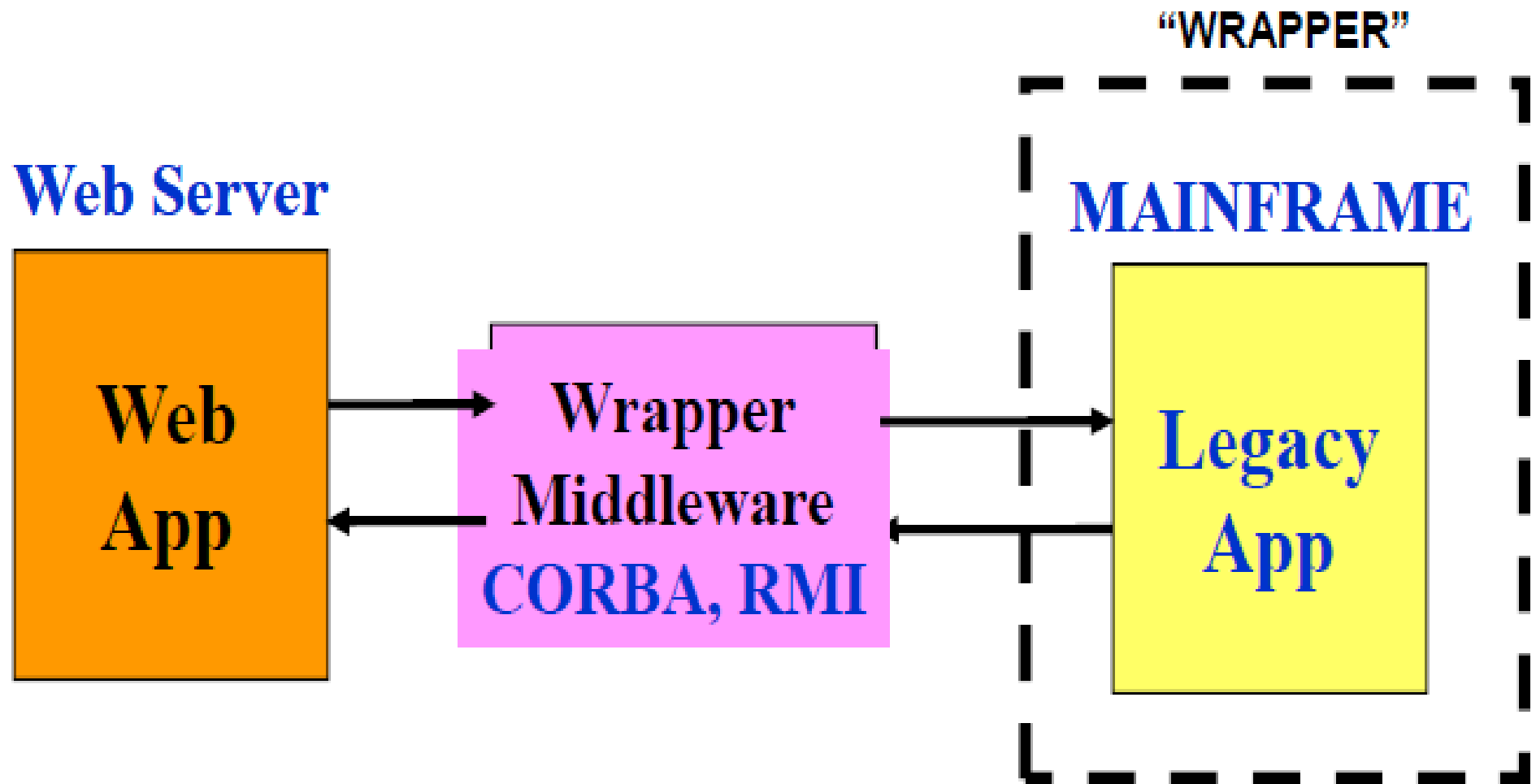


Web Architectures

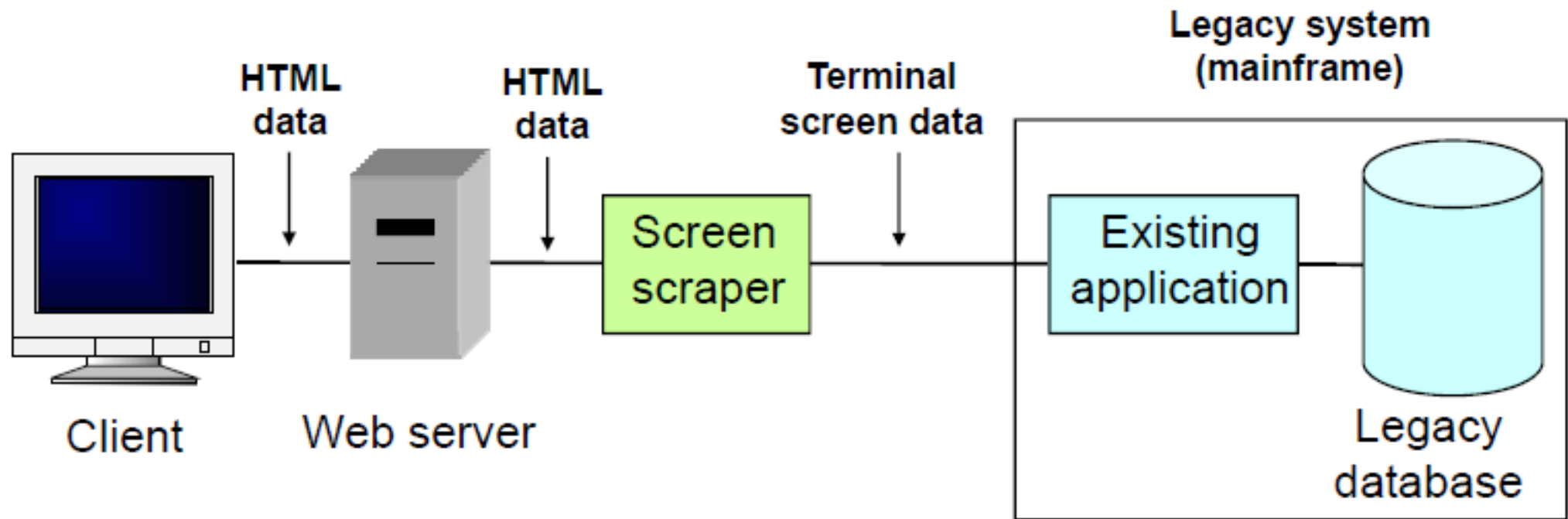
N-Tier Architecture



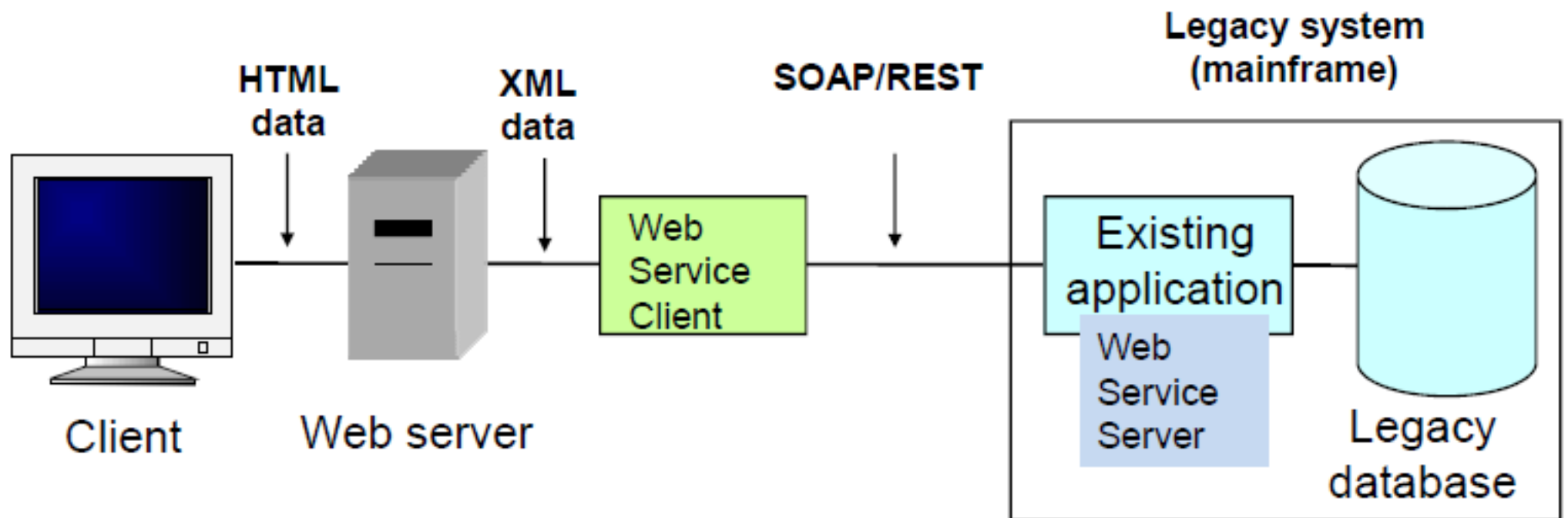
Connecting to Legacy System



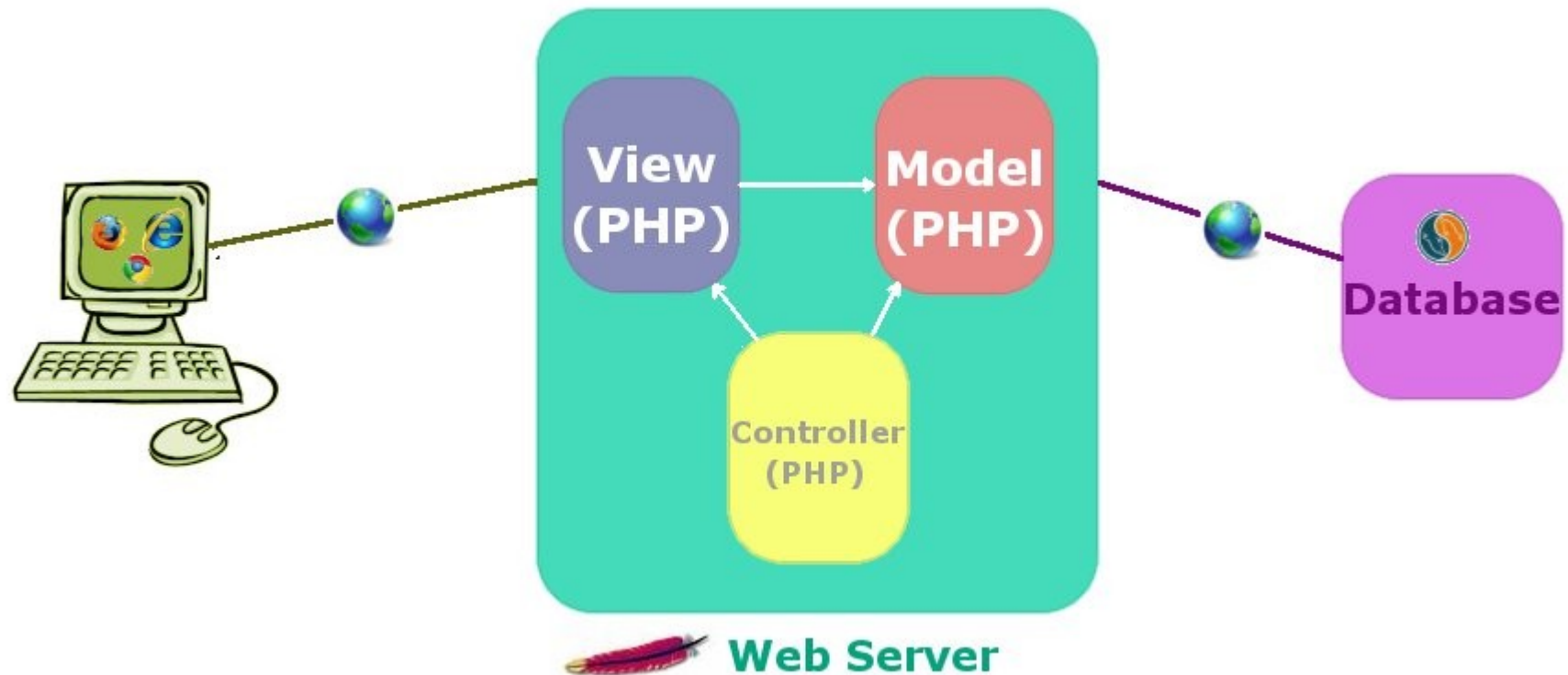
Connecting to Legacy System



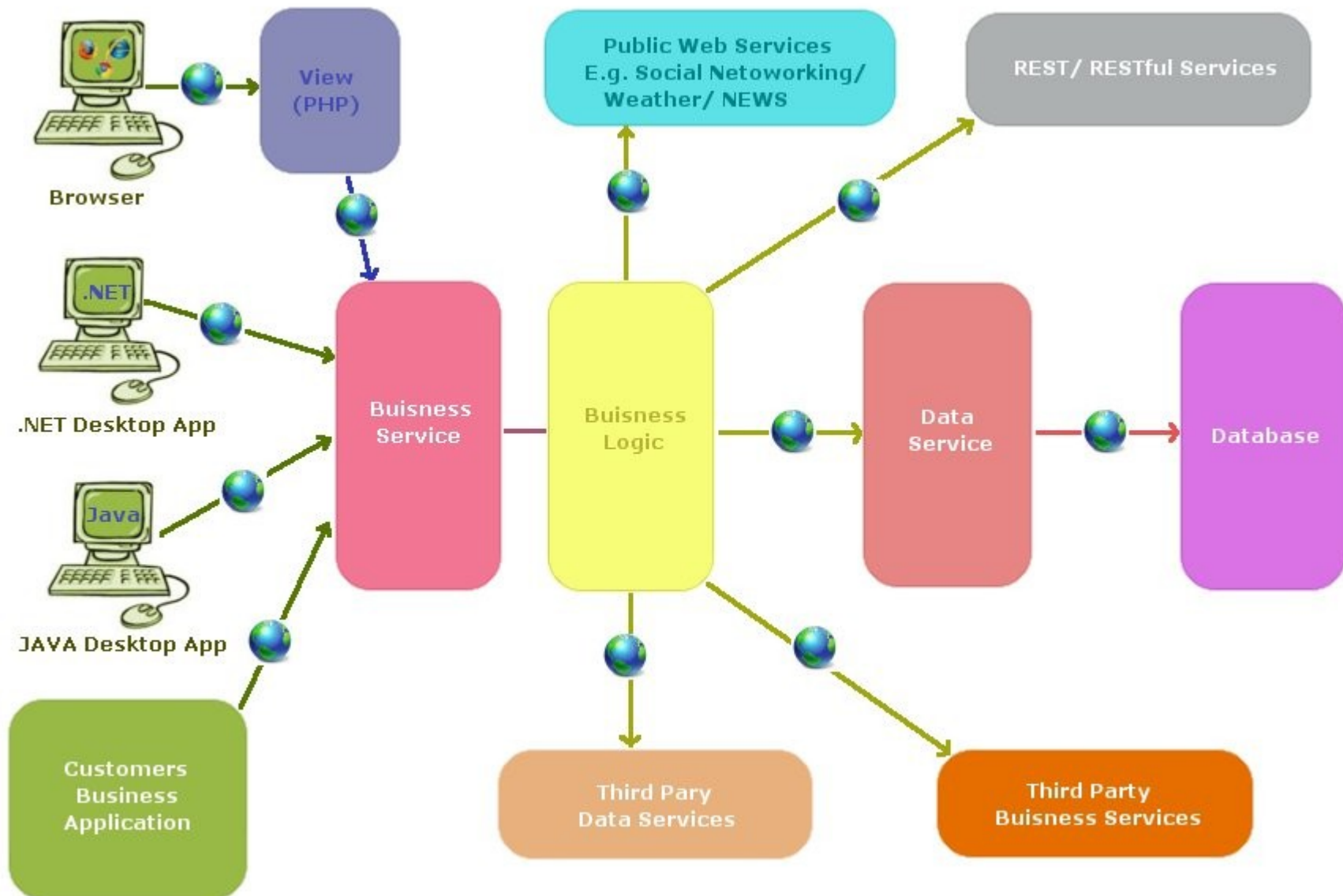
Connecting to Legacy System

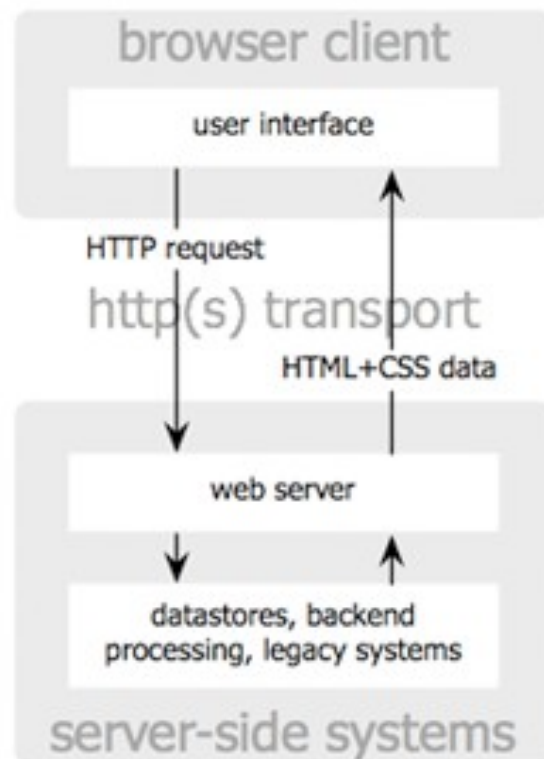


Traditional Web Application Architecture



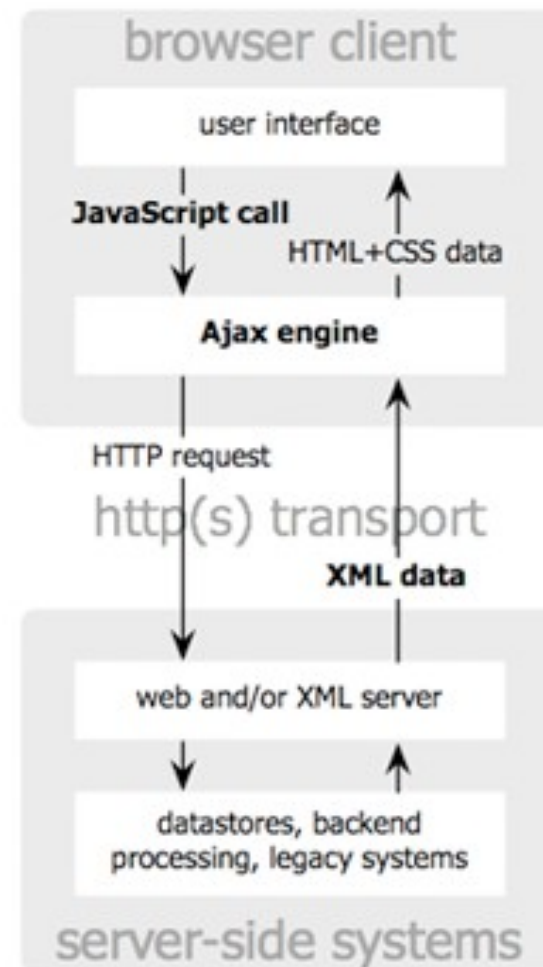
Current Web Application Architecture





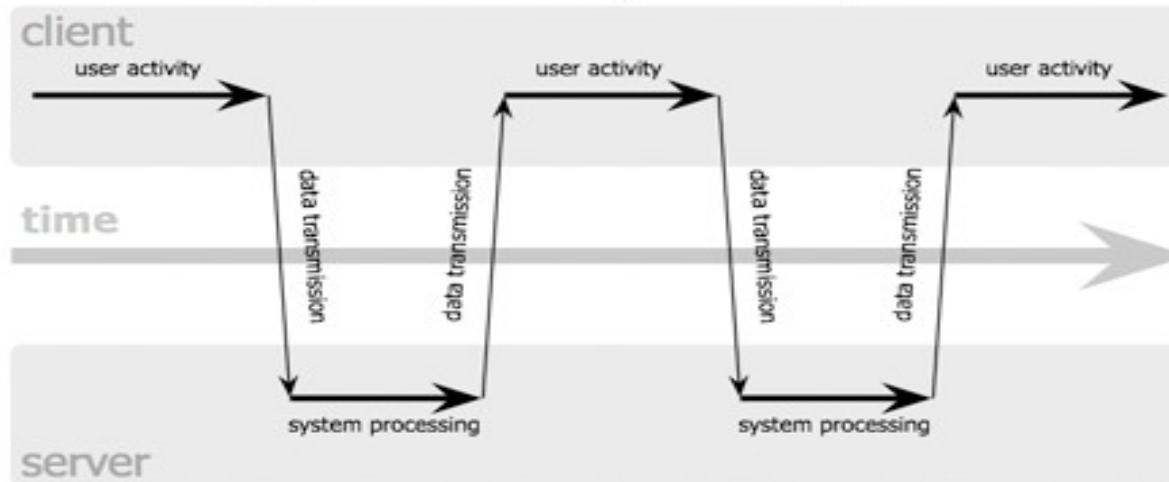
**classic
web application model**

Jesse James Garrett / adaptivepath.com

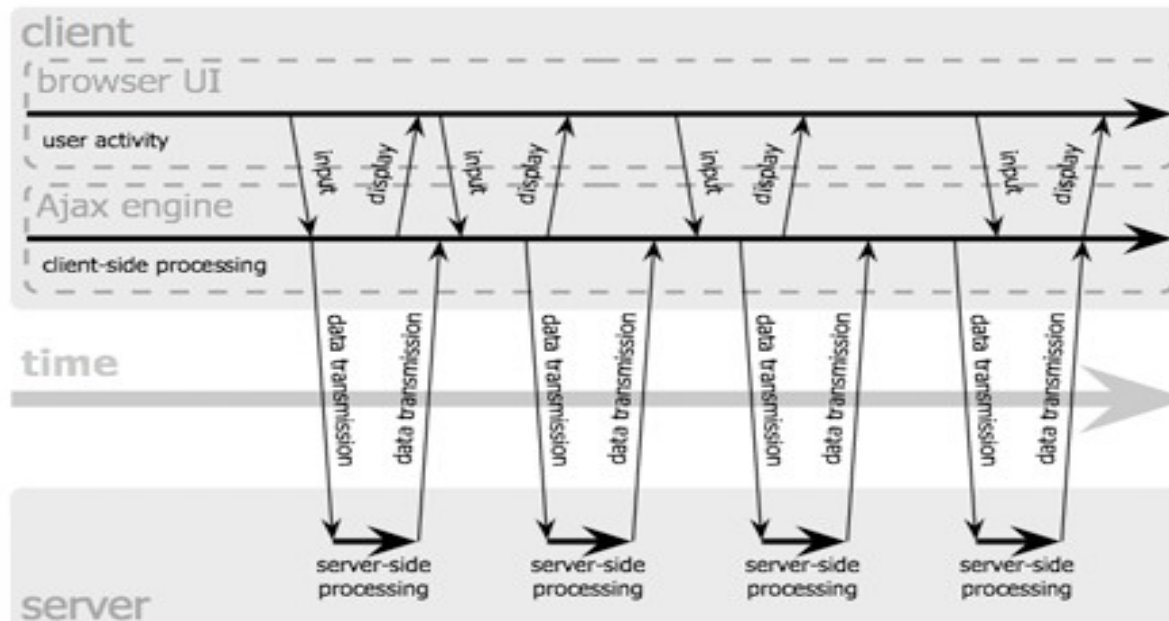


**Ajax
web application model**

classic web application model (synchronous)



Ajax web application model (asynchronous)



Web Applications

