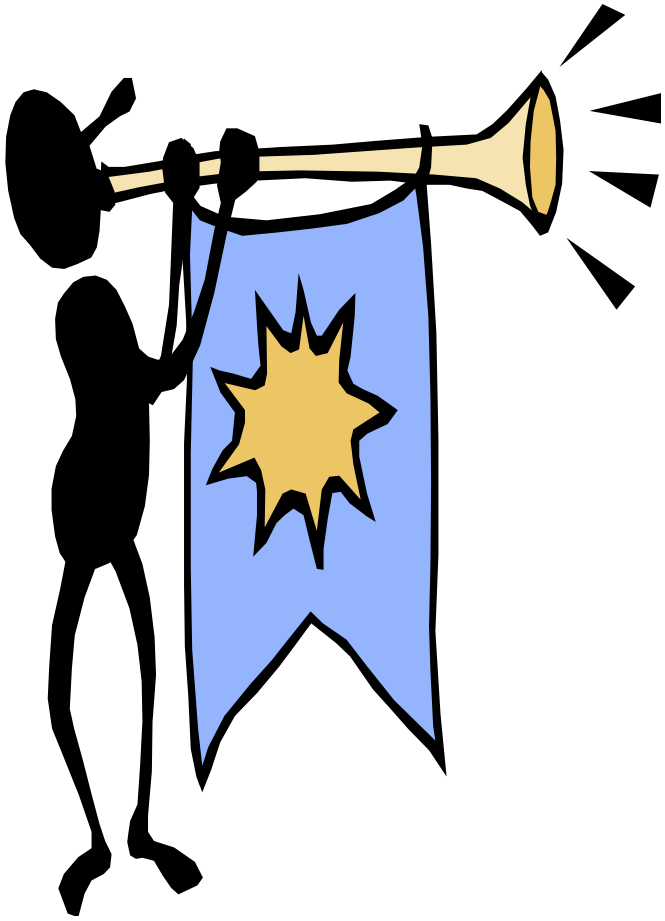
The World Wide Web

Application Layer

Application
Transport
Network
Data Link
Physical

Outline



- Introduction
- Architecture
- Documents
- HTTP

Introduction

World Wide Web – WWW

Is an architectural framework for accessing linked documents spread out over millions of machines all over the Internet

1989: Idea born at the European Centre for Nuclear Research (CERN) in Geneva

The Web grew out of the need to have large teams of internationally dispersed researchers collaborate (using reports, blueprints, drawings, photos, …)

1991: First public demonstration in San Antonio, Texas (Hypertext '91 conference)

Introduction

World Wide Web – WWW

1993: Release of the the first graphical browser (i.e. Mosaic) by University of Illinois

1994: Birth of the World Wide Web Consortium (W3C)

- Standardization body of the web (e.g. functional entities, protocols)

Introduction

World Wide Web – WWW

1995: Birth of the first product browsers

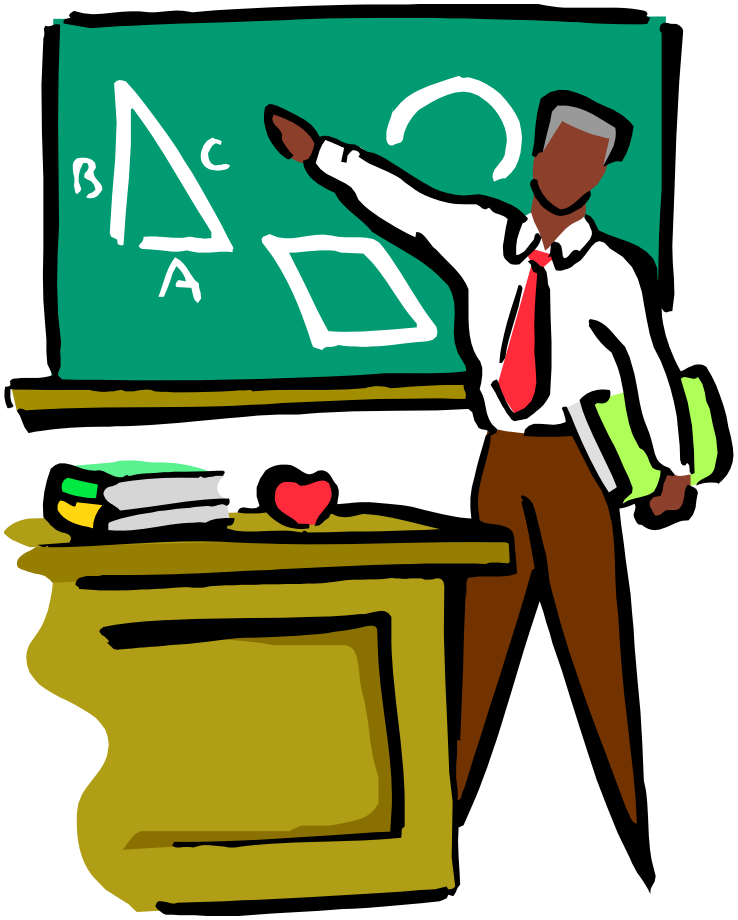
- Netscape – Utilise Mosaic comme base
- Microsoft Internet Explorer

1995 – 1998:

Rivalry between Netscape and Microsoft Explorer, which led to the demise of Netscape

After 1998: Many more browsers including open source browsers (e.g. Chrome – 2008)

Architecture



- 1. Overview
- 2. Browser
- 3. Web Server

Overview

- Client / server
  - Client side: browser
  - Server side: web pages
    - Universal Resource Locator (URL) used as page identifier
      - Ex: http://www.abc.com/products.html

Overview

Client / server

- Steps: client side

1. The browser determines the URL
2. The browser asks DNS for the IP address of the web server
3. The browser initiates a TCP connection with the web server, on port 80
4. The browser sends over an HTTP request

Overview

Client / server

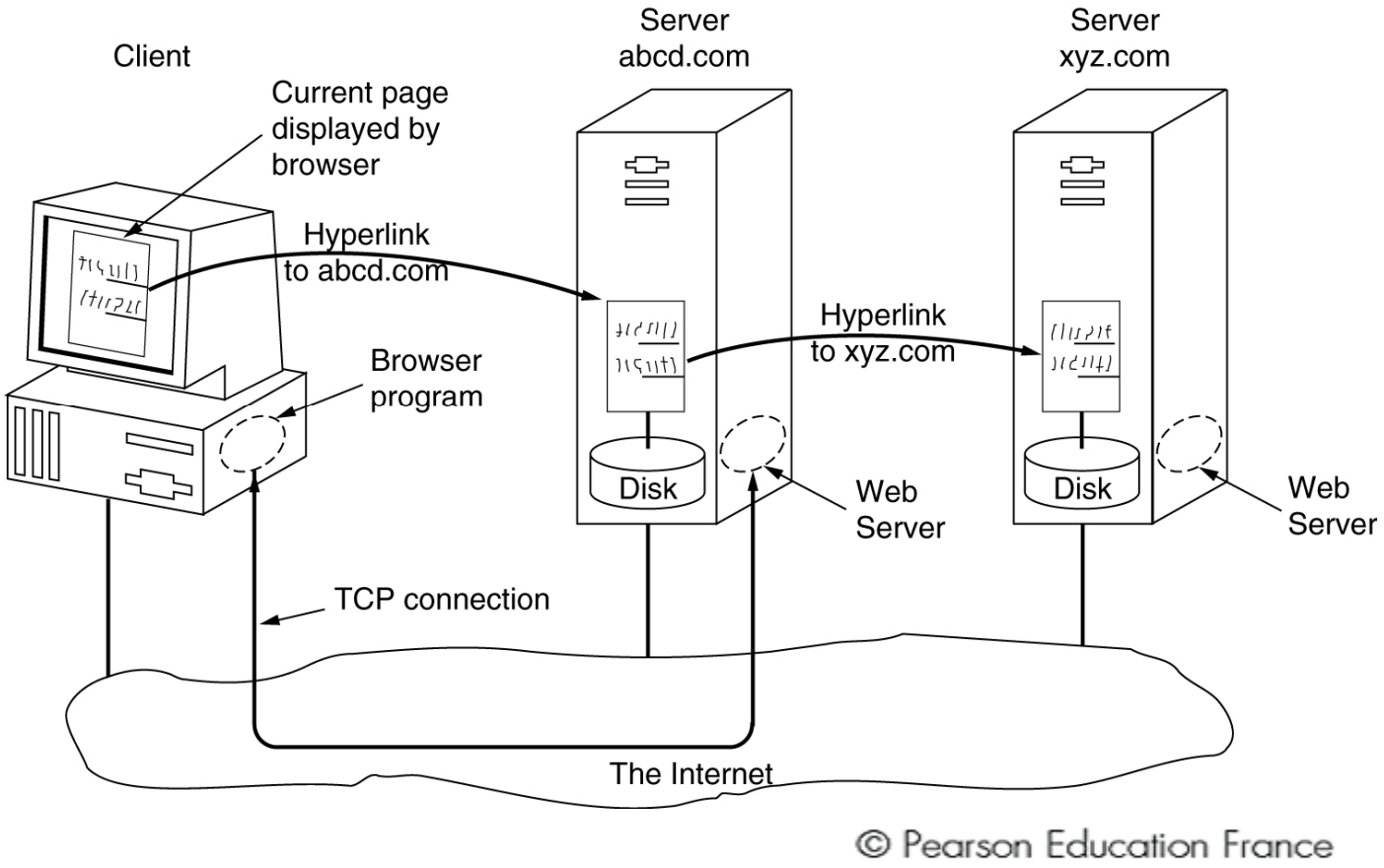
– Steps: server side

5. The server replies to the HTTP request, including the requested document
6. The TCP connection is released

- Client side

7. The browser displays the response

Overview



The parts of the Web model.

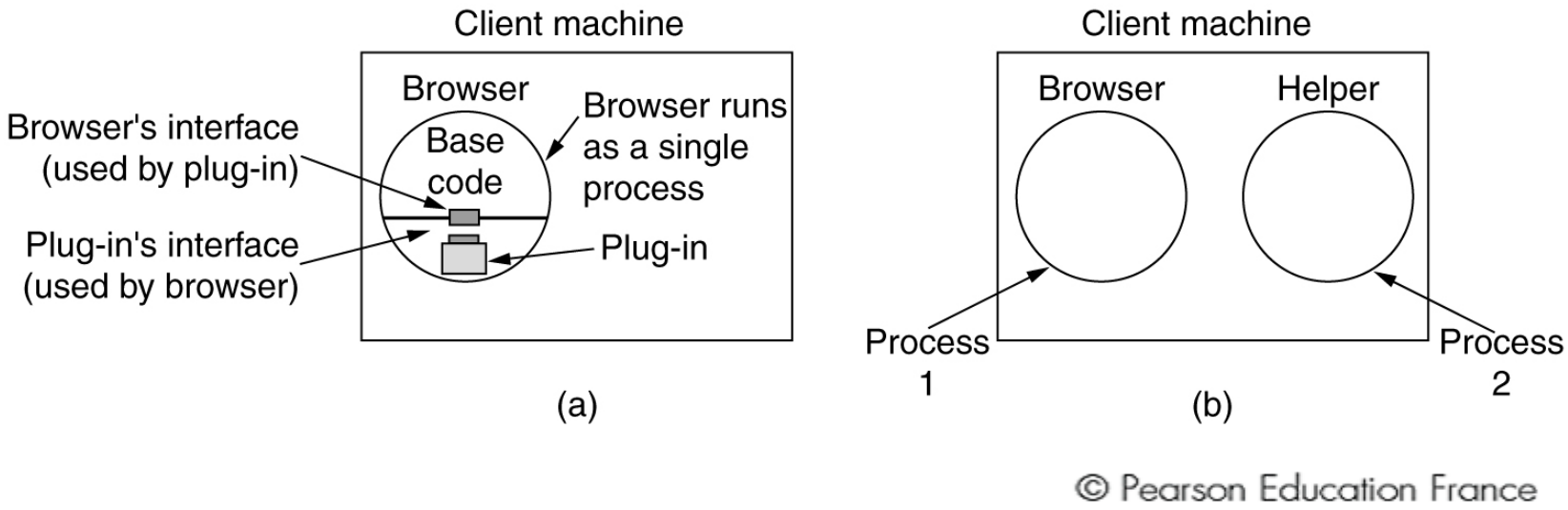
The browser

Two possibilities to extend the basic functionality

– Plug-in
  - A code module that the browser fetches from a special directory on the disk and installs as an extension to itself
  - Runs in the same process

- Helper application
  - A complete program, running as a separate process

The browser



(a) A browser plug-in.

(b) A helper application.

The Web server

Detailed tasks: Authentication, access control

1. Resolve the name of the Web page requested
2. Authenticate the client
3. Perform access control on the client
4. Perform access control on the Web page

The Web server

Detailed tasks: Find the requested page

5. Check the cache
6. Fetch the requested page from disk

The Web server

Detailed tasks: Prepare and send the response to the client

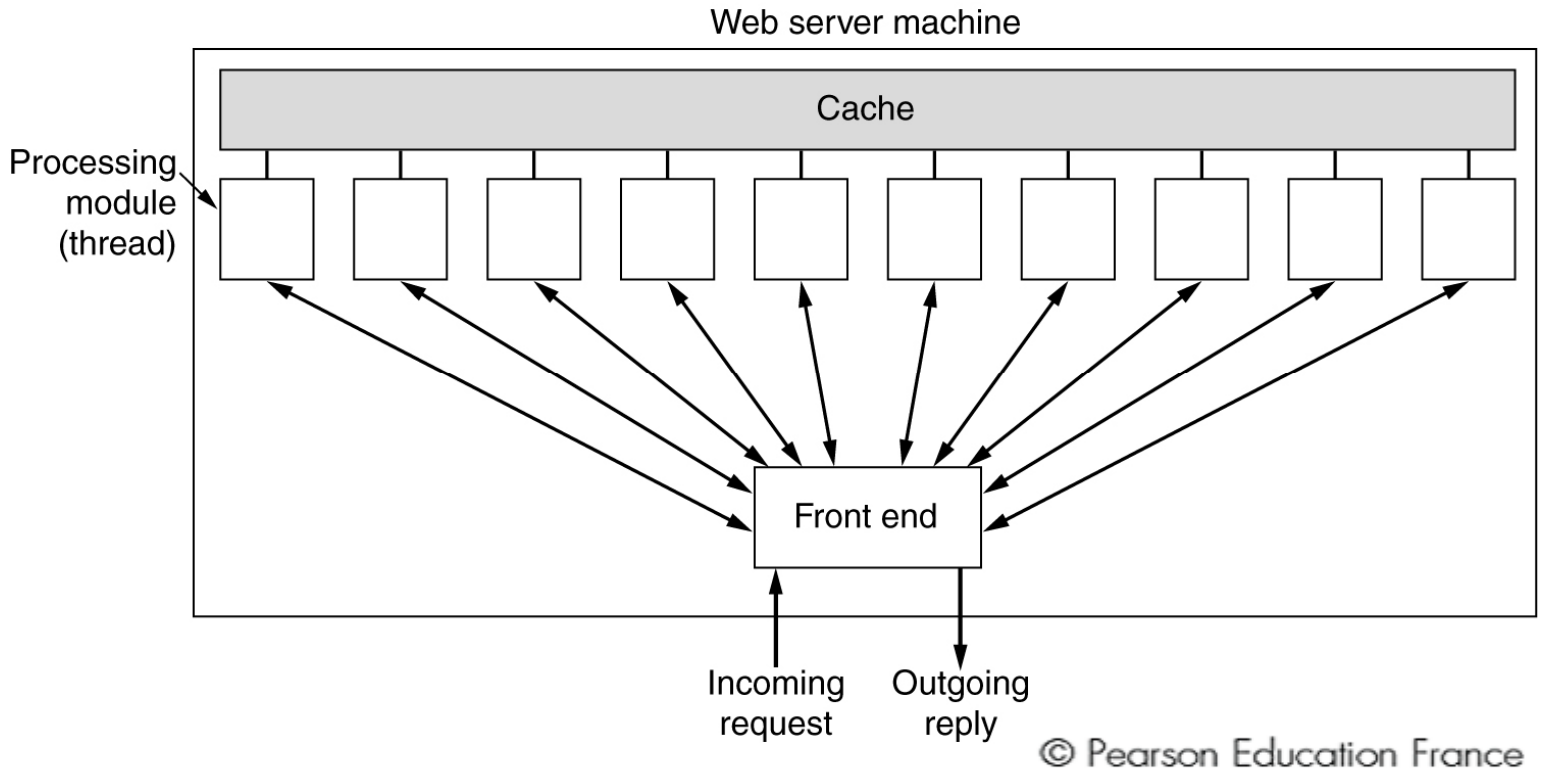
7. Determine the MIME type to include in the response.
8. Return the reply to the client.
9. Make an entry in the server log.

The Web server

To enhance the server performances

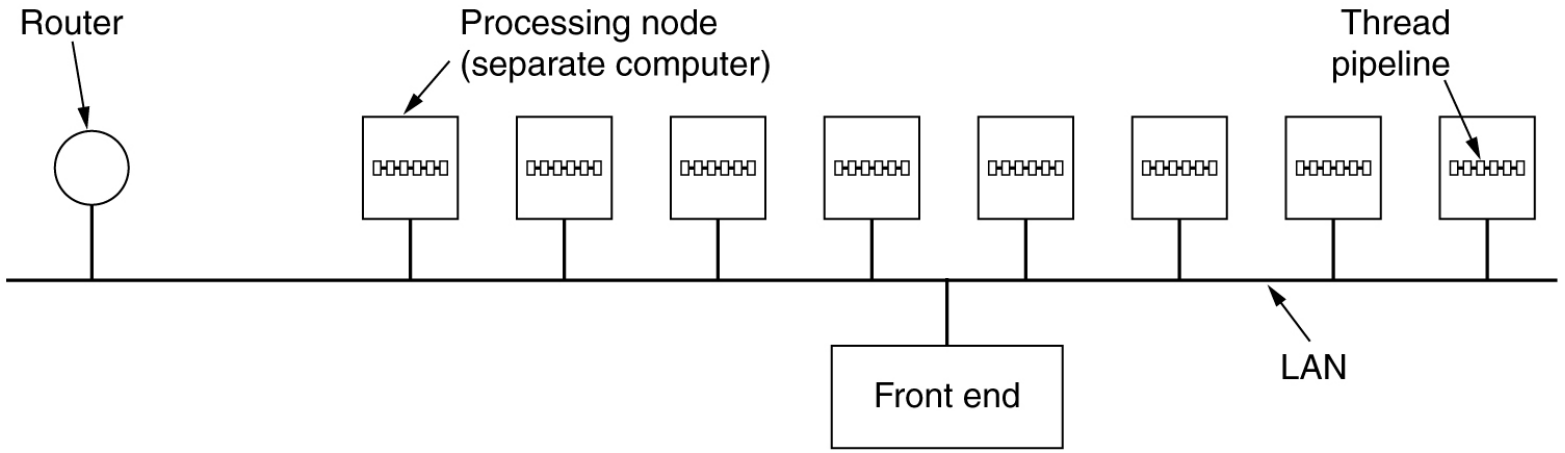
1. Cache
2. Multithreading
3. Server farm

The Web server



A multithreaded Web server with a front end and processing modules.

The Web server



A server farm.

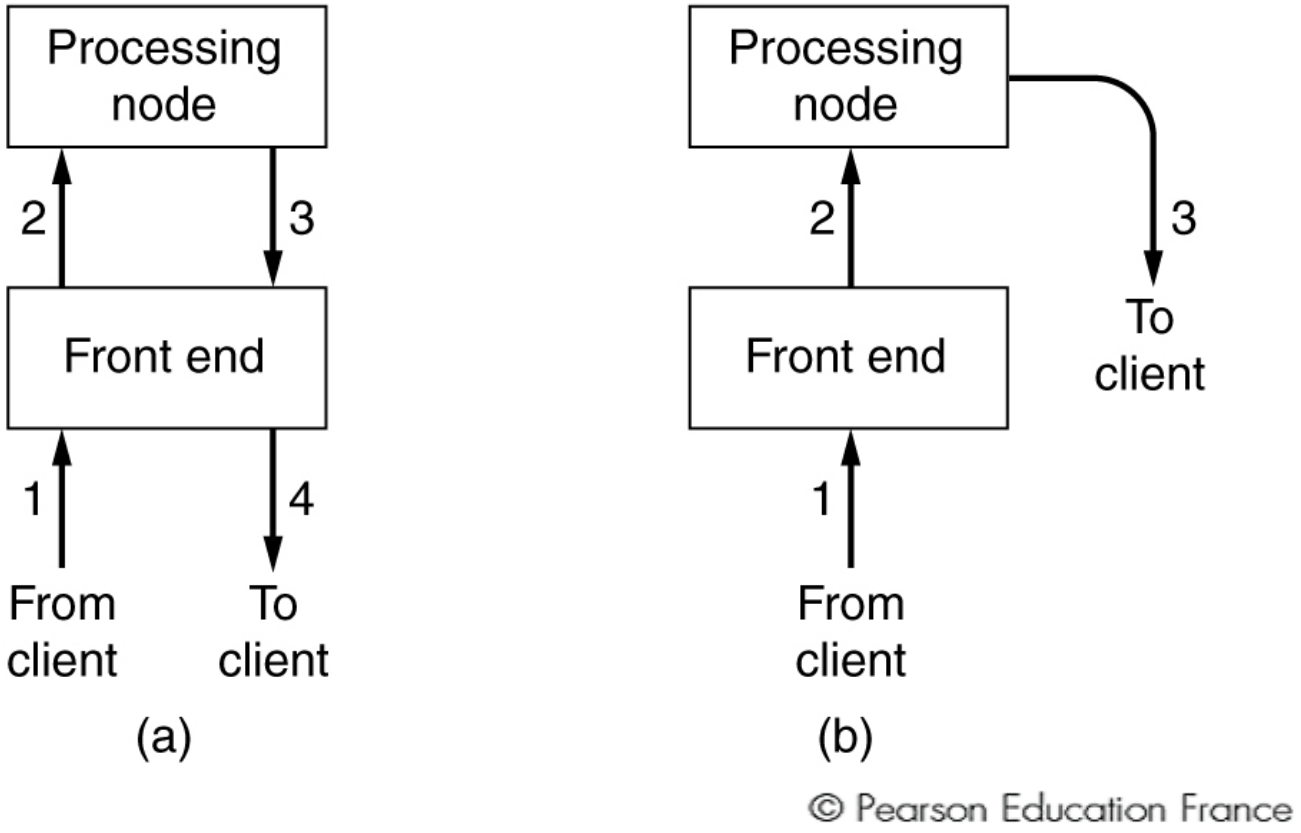
The Web server

TCP Connexion

Two possibilities

1. The client's TCP connection terminates at the front end
   - The reply must also go through the front end)
2. TCP handoff
   - the TCP end point is passed to the processing node so it can reply directly to the client,

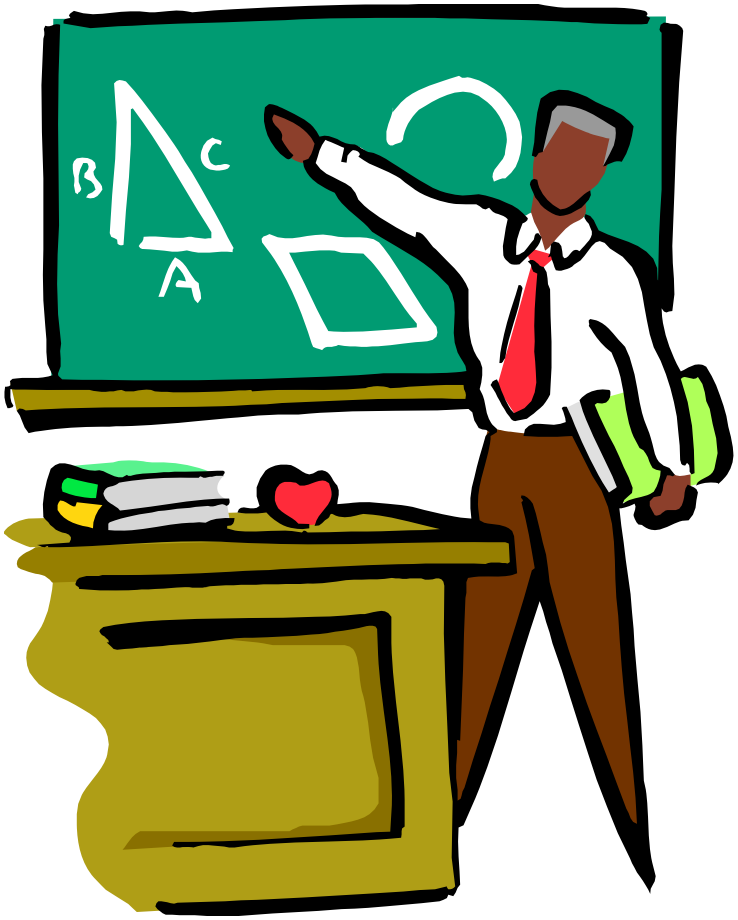
The Web server



(a) Normal request-reply message sequence.

(b) Sequence when TCP handoff is used.

Documents



- 1. Identification
- 2. Static Web documents
- 3. Dynamic Web documents

Identification

Identifier = URL (Uniform Resource Locators)

Essential questions:

1. What is the page called?
2. Where is the page located?
3. How can the page be accessed?

Identification

URL

Answers

1. What is the page called?
   - A local name uniquely indicating the specific page (the file name)
2. Where is the page located?
   - DNS name of the machine on which the page is located
3. How can the page be accessed?
   - The protocol used (ex: HTTP, FTP)

– E.g. http://www. Example.com/video/index.html

Identification

Some common URLs.

Name	Used for	Example
http	Hypertext (HTML)	http://www.cs.vu.nl/~ast/
ftp	FTP	ftp://ftp.cs.vu.nl/pub/minix/README
file	Local file	file:///usr/suzanne/prog.c
news	Newsgroup	news:comp.os.minix
news	News article	news:AA0134223112@cs.utah.edu
gopher	Gopher	gopher://gopher.tc.umn.edu/11/Libraries
mailto	Sending e-mail	mailto:JohnUser@acm.org
telnet	Remote login	telnet://www.w3.org:80

Static Web documents

Static web pages are written in languages such as

- HTML (Hypertext Mark up Language).
- XML

Static Web documents

HTML (Hypertext Mark up Language)

- Problems
  - Mixes content with formatting instructions
  - Do not allow to structure documents

Static Web documents

- XML is a markup language for documents containing structured information

- XML makes use of tags just like HTML.
  - In HTML, both tag semantics (<p> means paragraph) and tag set are fixed

W3C recommendation

Static Web documents

Difference between XML and HTML

- XML was designed to carry data
- XML is not a replacement for HTML
- XML and HTML were designed with different goals
  - XML was designed to describe data and to focus on what data is
  - HTML was designed to display data and to focus on how data looks.
- HTML is about displaying information, while XML is about describing information.
- XML is free and extensible (xml tags are not predefined)

Static Web documents

XML documents Example

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<book>
    <title>Understanding Web Services</title>
    <author>Eric Newcomer</author>
    <price>39.99</price>
</book>
```

Static Web documents

XML processor

- Read XML documents
- Provide access to the content and the structure
- Navigate XML document structure and add, modify, or delete its elements.

- Most popular programming APIs
  - Document Object Model (DOM) from W3C
  - Simple API for XML (SAX) – From XML-DEV mailing list

Dynamic Web documents

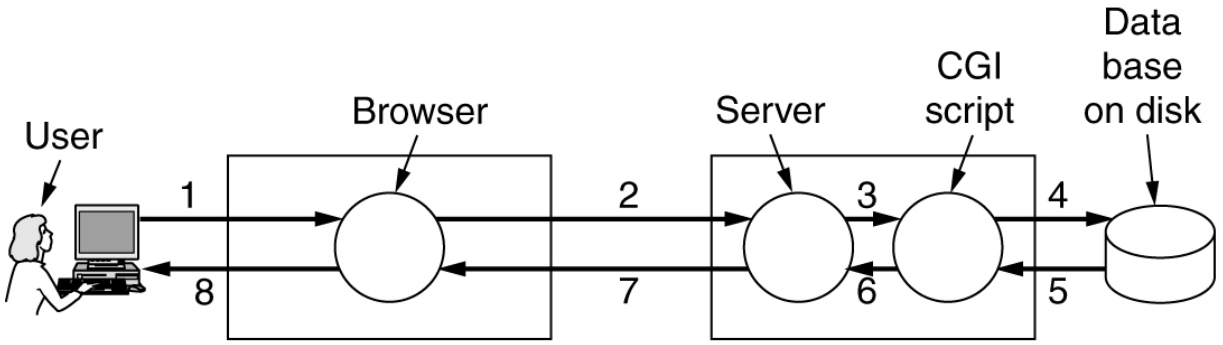
Why we need dynamic content?

- Look up a record in a database
- Return a page that needs to be generated on-the-fly
  - Ex: Number of the web site visitors

How to provide dynamic content?

- Use scripts (e.g. CGI-Common Gateway Interface)
- Use JAVA Servlets

Dynamic documents



1. User fills in form
2. Form sent back
3. Handed to CGI
4. CGI queries DB
5. Record found
6. CGI builds page
7. Page returned
8. Page displayed

Steps in processing the information from an HTML form

Dynamic Web documents

Shortcomings of the CGI script approach

- The script must run on the Web server (can cause scalability problems)
- The databases must be re-opened and re-closed for each request (performance issues)

Servlet API

Creation of dynamic Web content

- Servlet
  - Java component
  - Generate content on the fly, just like CGI
    - interface between HTTP request and data bases
    - Forms
    - Dynamic information (e.g. date, number of visitors)

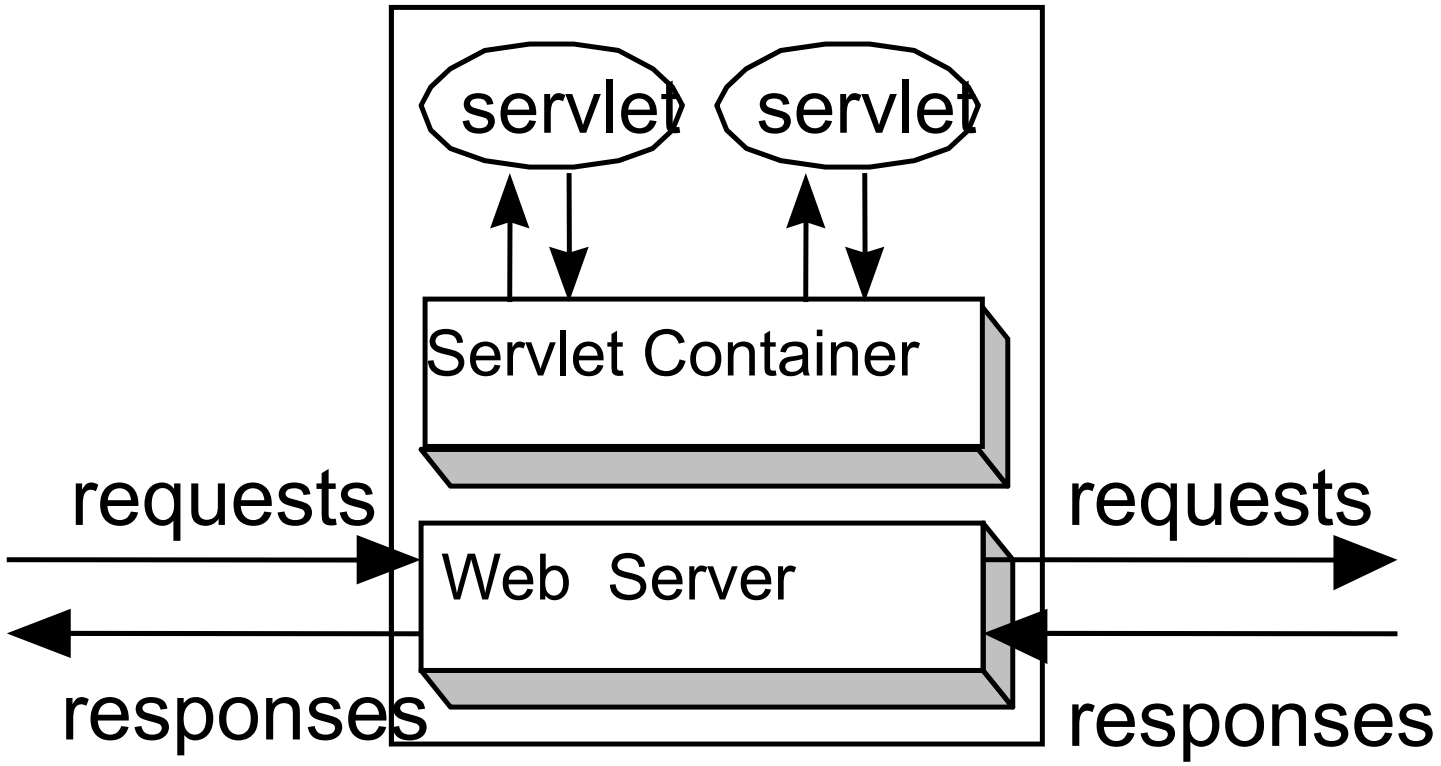
Servlet API

Servlet container (also know as servlet engine)

- Contains the servlets
- Manages the servlets through their life cycle
  - Creation
  - Initialisation
  - Destruction
  - Receives and decodes of HTTP requests
  - Encodes and sends of HTTP responses

Servlet container

A container collocated with a proxy server



Dynamic Web documents

Other methods besides CGI and Servlets

1. PHP: Hypertext preprocessor

- The PHP script is embeded inside HTML pages and it is executed by the server itself to generate the page
- PHP script tag
  <?php ?>

Dynamic Web documents

Example

```
<html>
<body>

<h2> This is what I know about you </h2>
<?php echo $HTTP_USER_AGENT ?>

</body>
</html>
```

Dynamic Web documents

```
<html>
<body>
<form action="action.php" method="post">
<p> Please enter your name: <input type="text" name="name"> </p>
<p> Please enter your age: <input type="text" name="age"> </p>
<input type="submit">
</form>
</body>
</html>
```

(a)

```
<html>
<body>
<h1> Reply: </h1>
Hello <?php echo $name; ?>.
Prediction: next year you will be <?php echo $age + 1; ?>
</body>
</html>
```

(b)

```
<html>
<body>
<h1> Reply: </h1>
Hello Barbara.
Prediction: next year you will be 25
</body>
</html>
```

(c)

(a) A Web page containing a form. (b) A PHP script for handling the output of the form. (c) Output from the PHP script when the inputs are "Barbara" and 24 respectively.

Dynamic Web documents

Other methods besides CGI and Servlets

2. Java server page (JSP)

- Similar to PHP but written in Java

Dynamic Web documents

Other methods besides CGI and Servlets

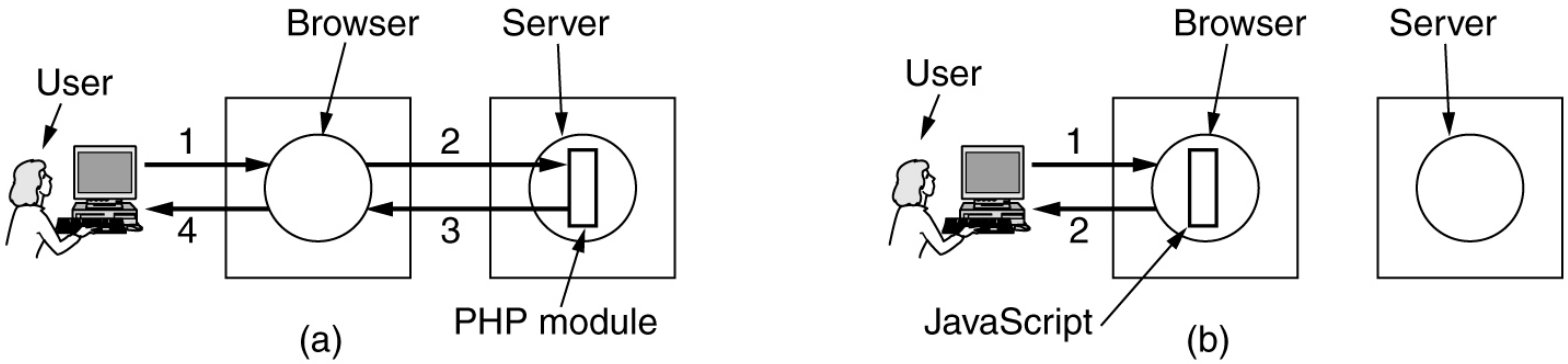
3. Active Server Page
   - Microsoft version, written using Visual basic

Dynamic Web documents

Dynamic content can also be generated on the client side

1. Java script
2. Applets

Dynamic Web documents



(a) Server-side scripting with PHP.
(b) Client-side scripting with JavaScript.

Dynamic Web documents

Use of JavaScript for processing a form.

```
<head>
<script language="javascript" type="text/javascript">
function response(test  form) {
    var person = test  form.name.value;
    var years = eval(test  form.age.value) + 1;
    document.open();
    document.writeln("<html> <body>");
    document.writeln("Hello " + person + ".<br>");
    document.writeln("Prediction: next year you will be " + years + ".");
    document.writeln("</body> </html>");
    document.close();
}
</script>
</head>

<body>
<form>
Please enter your name: <input type="text" name="name">
<p>
Please enter your age: <input type="text" name="age">
<p>
<input type="button" value="submit" onclick="response(this.form)">
</form>
</body>
</html>
```

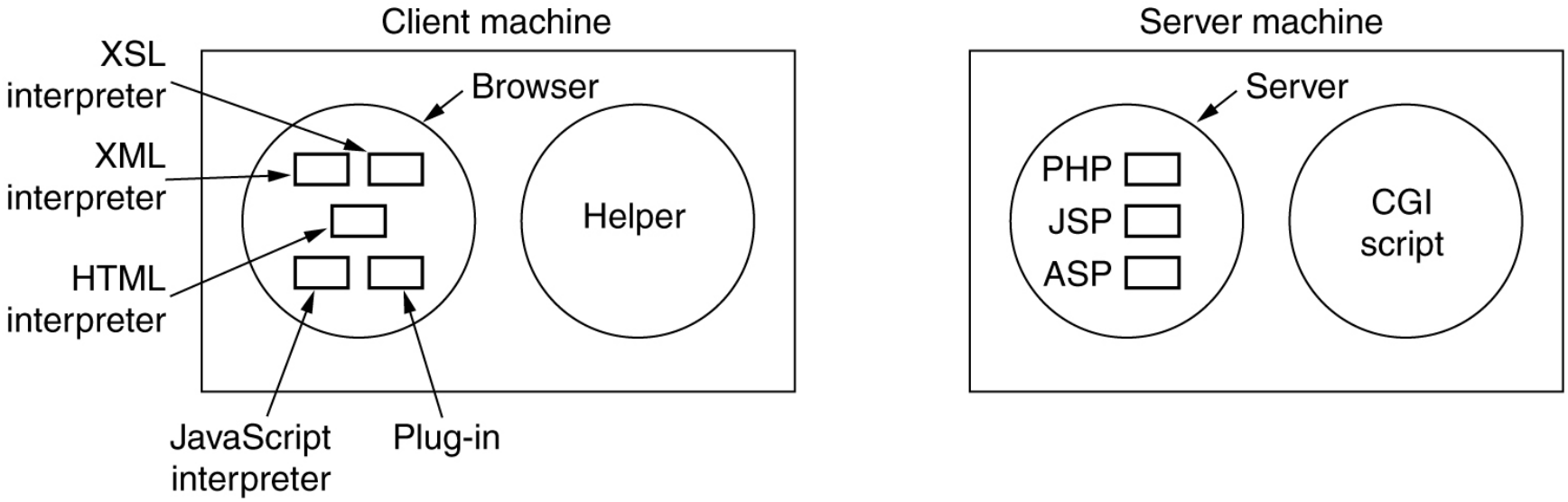
Dynamic Web documents

```
<html>
<head>
<script language="javascript" type="text/javascript">

function response(test_form) {
   function factorial(n) {if (n == 0) return 1; else return n * factorial(n – 1);}
   var r = eval(test_form.number.value);       //  r = typed in argument
   document.myform.mytext.value = "Here are the results.\n";
   for (var i = 1; i <= r; i++)                          // print one line from 1 to r
      document.myform.mytext.value += (i + "! =  " + factorial(i) + "\n");
}
</script>
</head>

<body>
<form name="myform">
Please enter a number: <input type="text" name="number">
<input type="button" value="compute table of factorials" onclick="response(this.form)">
<p>
<textarea name="mytext" rows=25 cols=50> </textarea>
</form>
</body>
</html>
```

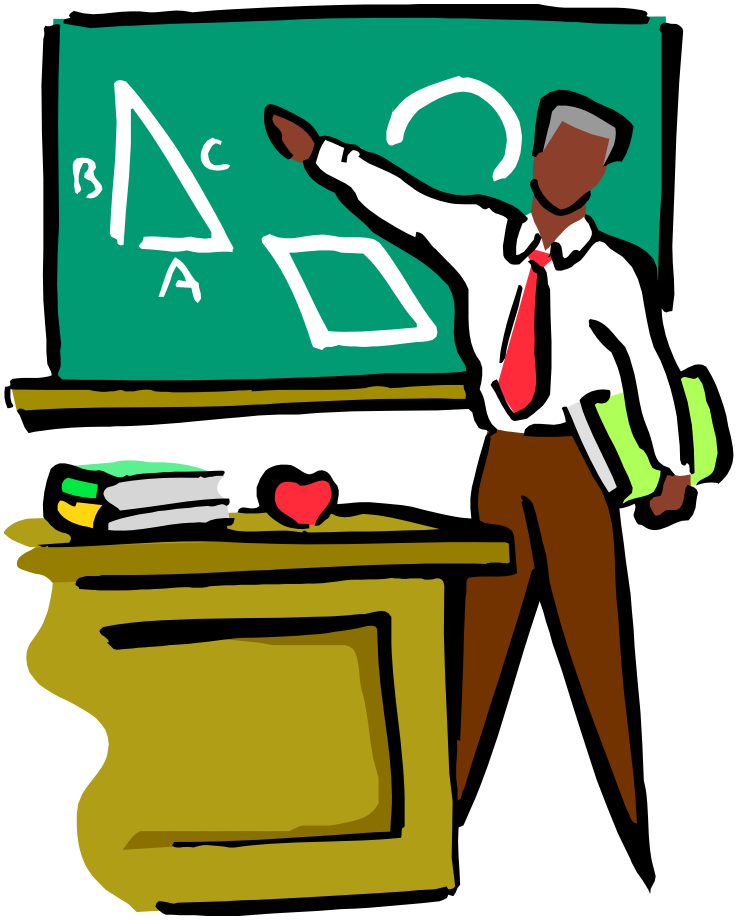
A JavaScript program for computing and printing factorials.

Dynamic Web documents

Summary



HTTP v1



- 1. Introduction
- 2. Functional entities
- 3. Messages
- 4. Methods

Introduction

HTTP (HyperText Transfer Protocol)

- Is an application-level protocol for distributed, collaborative, hypermedia information systems
  - HTTP has been in use since 1990
  - HTTP is a request-response protocol
  - HTTP requests relates to resources
    - A resource is any object or service network that can be identified by a URI (Universal Resource Identifier)

Functional entities

Client

- A program that establishes connections for the purpose of sending requests

User Agent

- The client which initiates a request (e.g. browser)

- Note
  - A request may pass through several servers

Functional entities

Server

- An application program that accepts connections in order to service requests by sending back responses
- A given program may be capable of being both a client and a server
- The role depends on connections

Functional entities

- **Origin server**
  - The server on which a given resource resides or is to be created
- **Proxy server**
  - An intermediary program which acts as both a server and a client for the purpose of making requests on behalf of other clients
- **Gateway server**
  - receives requests as if it were the origin server for the requested resource, and forwards the request to another server
  - Is transparent to the client

Messages

HTTP-message = Request | Response

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

start-line = Request-Line | Status-Line

Methods (requests)

HEAD

- retrieve meta-information about a web page, without retrieving the page content (ex: get the date for last modification)

GET

- retrieve the page content

PUT

- store the enclosed content under the supplied Request-URI

POST

- add the entity enclosed in the request as a new subordinate of the resource identified by the Request-URI
  - E.g.
    - Post a message to a mailinglist
    - Extend a database by appending information
    - Transfer a form data

Methods (requests)

DELETE

- Deletes the page

TRACE

- Debug

OPTIONS

- Allows the client to discover the options supported by the server supporte

CONNECT

- Not used currently

Methods (Requests)

The built-in HTTP request methods.

Method	Description
GET	Request to read a Web page
HEAD	Request to read a Web page's header
PUT	Request to store a Web page
POST	Append to a named resource (e.g., a Web page)
DELETE	Remove the Web page
TRACE	Echo the incoming request
CONNECT	Reserved for future use
OPTIONS	Query certain options

Methods (responses)

The status code response groups.

Code	Meaning	Examples
1xx	Information	100 = server agrees to handle client's request
2xx	Success	200 = request succeeded; 204 = no content present
3xx	Redirection	301 = page moved; 304 = cached page still valid
4xx	Client error	403 = forbidden page; 404 = page not found
5xx	Server error	500 = internal server error; 503 = try again later

Headers

Header	Type	Contents
User-Agent	Request	Information about the browser and its platform
Accept	Request	The type of pages the client can handle
Accept-Charset	Request	The character sets that are acceptable to the client
Accept-Encoding	Request	The page encodings the client can handle
Accept-Language	Request	The natural languages the client can handle
Host	Request	The server's DNS name
Authorization	Request	A list of the client's credentials
Cookie	Request	Sends a previously set cookie back to the server
Date	Both	Date and time the message was sent
Upgrade	Both	The protocol the sender wants to switch to
Server	Response	Information about the server
Content-Encoding	Response	How the content is encoded (e.g., gzip)
Content-Language	Response	The natural language used in the page
Content-Length	Response	The page's length in bytes
Content-Type	Response	The page's MIME type
Last-Modified	Response	Time and date the page was last changed
Location	Response	A command to the client to send its request elsewhere
Accept-Ranges	Response	The server will accept byte range requests
Set-Cookie	Response	The server wants the client to save a cookie

Some HTTP message headers.

References

1. A. Tanenbaum, Computer Networks, Seventh Edition,

1. http://www.w3schools.com/html/html_intro.asp

1. HTTP RFC (i.e. RFC 2616)