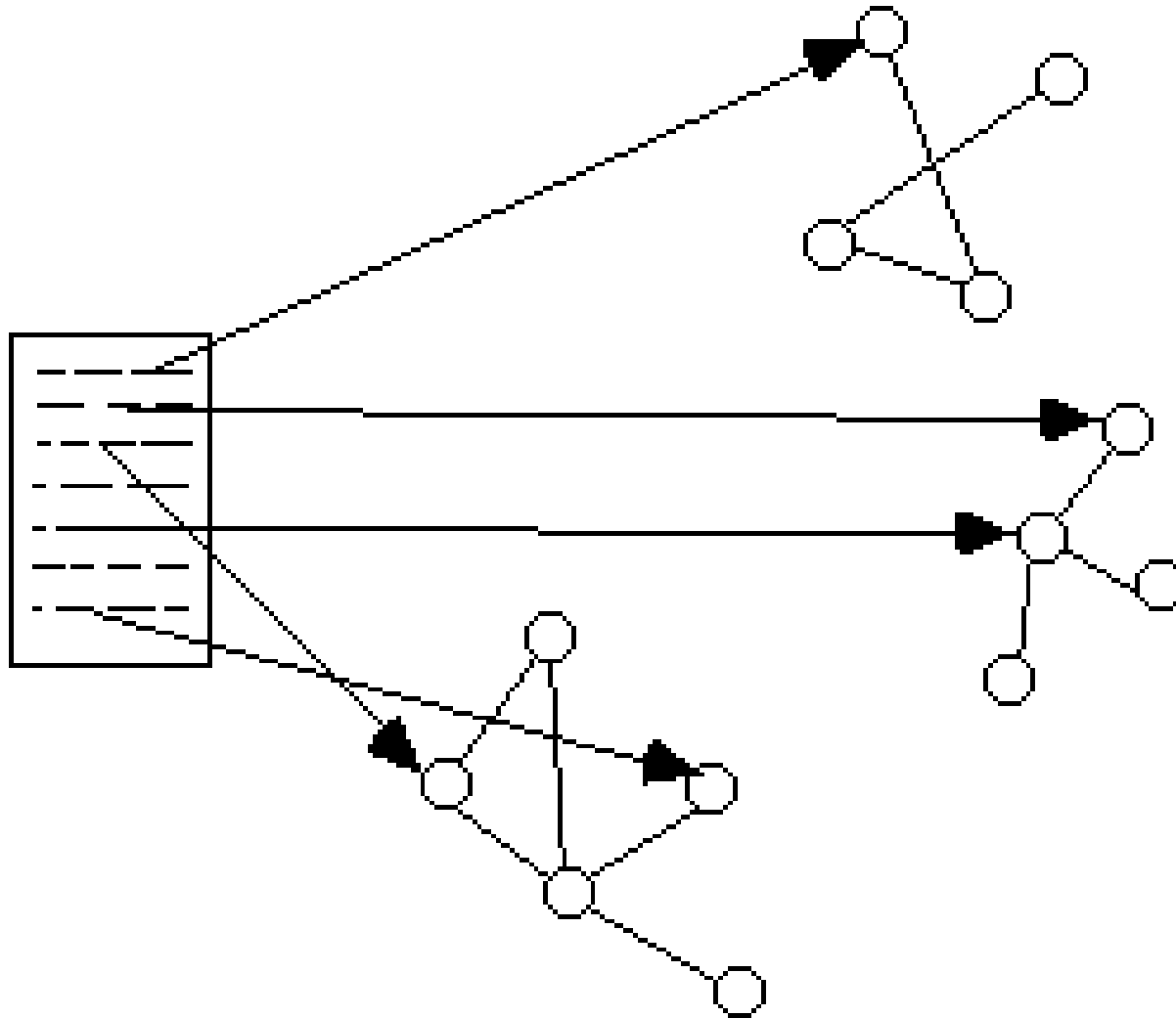


# ICS 362 Distributed Systems

***Distributed Systems: Part 17***

***Lecturer: Toby Daniel***

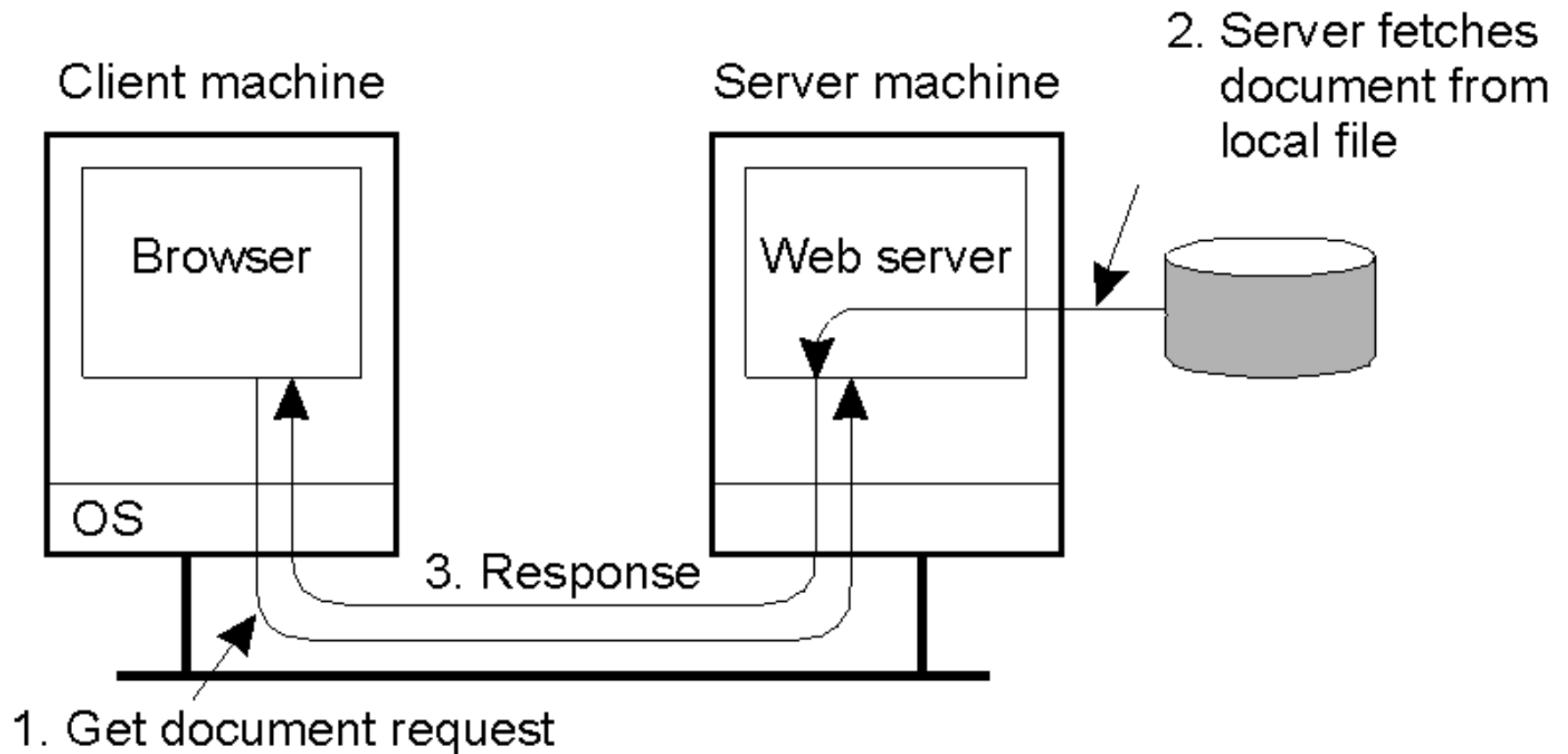
# Document Based Distributed Systems



# Document Based Distributed Systems

- During this lecture we are going to consider the WWW as a case study of a DS document based system.

# WWW Introduction



# Document Model 1

```
<HTML>                                     <!-- Start of HTML document -->
<BODY>                                     <!-- Start of the main body -->
<H1>Hello World/</H1>                     <!-- Basic text to be displayed -->
<P>                                       <!-- Start of a new paragraph -->
<SCRIPT type = "text/javascript">        <!-- identify scripting language -->
  document.writeln ("<H1>Hello World</H1>; // Write a line of text
</SCRIPT>                                <!-- End of scripting section -->
</P>                                     <!-- End of paragraph section -->
</BODY>                                  <!-- End of main body -->
</HTML>                                  <!-- End of HTML section -->
```

- A simple HTML based document, including Javascript.

# Document Model 2

- (1) <!ELEMENT article (title, author+,journal)>
- (2) <!ELEMENT title (#PCDATA)>
- (3) <!ELEMENT author (name, affiliation?)>
- (4) <!ELEMENT name (#PCDATA)>
- (5) <!ELEMENT affiliation (#PCDATA)>
- (6) <!ELEMENT journal (jname, volume, number?, month? pages, year)>
- (7) <!ELEMENT jname (#PCDATA)>
- (8) <!ELEMENT volume (#PCDATA)>
- (9) <!ELEMENT number (#PCDATA)>
- (10) <!ELEMENT month (#PCDATA)>
- (11) <!ELEMENT pages (#PCDATA)>
- (12) <!ELEMENT year (#PCDATA)>

- An XML document definition referring to a journal article.

# Document Model 3

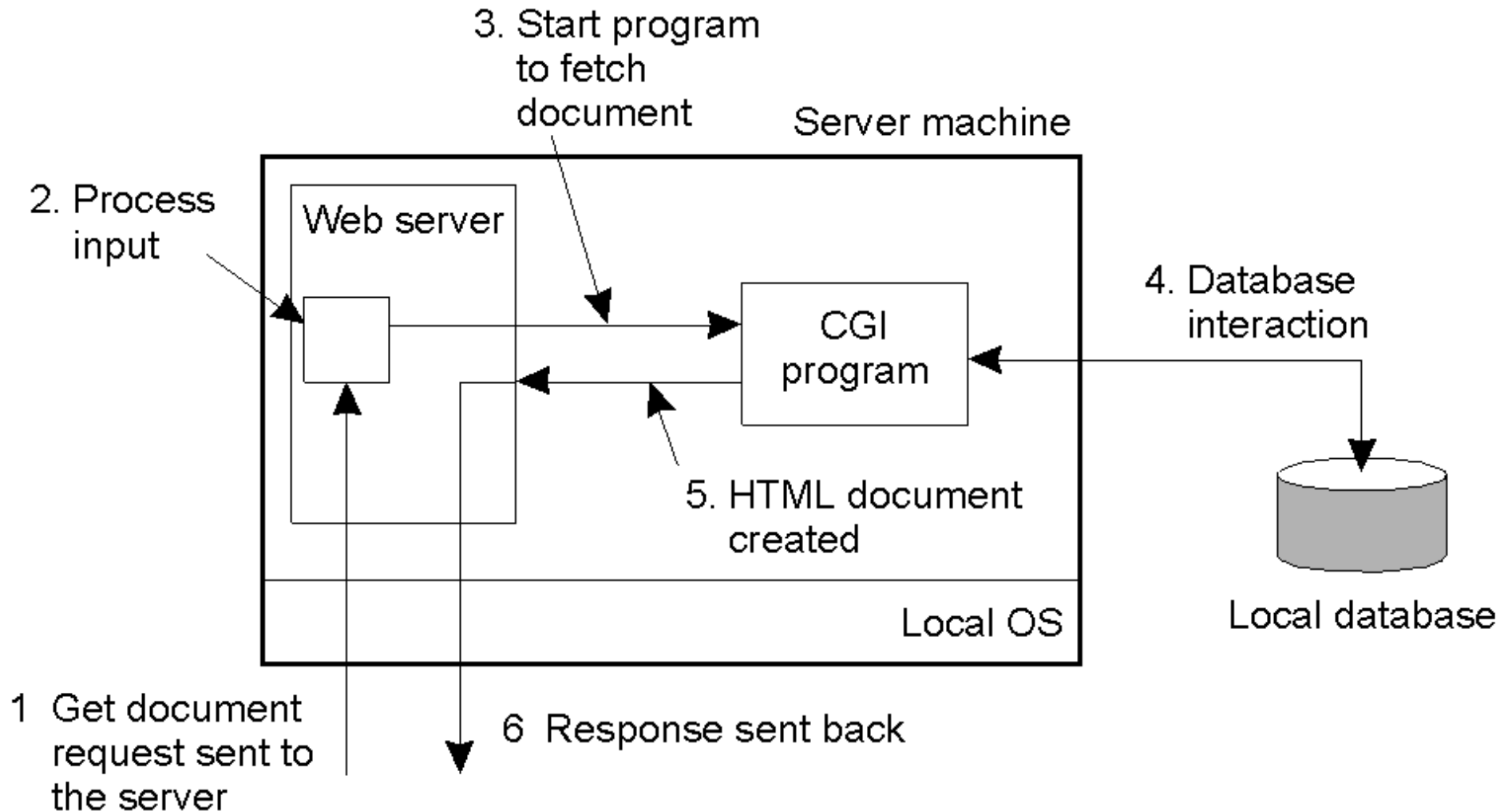
```
(1) <?xml = version "1.0">
(2) <!DOCTYPE article SYSTEM "article.dtd">
(3) <article>
(4)   <title> Prudent Engineering Practice for Cryptographic Protocols</title>
(5)   <author><name>M. Abadi</name></author>
(6)   <author><name>R. Needham</name></author>
(7)   <journal>
(8)     <jname>IEEE Transactions on Software Engineering</jname>
(9)     <volume>22</volume>
(10)    <number>12</number>
(11)    <month>January</month>
(12)    <pages>6 – 15</pages>
(13)    <year>1996</year>
(14)  </journal>
(15) </article>
```

- An XML document including definitions from previous slide.

# Further Document Types

Type	Subtype	Description
Text	Plain	Unformatted text
	HTML	Text including HTML markup commands
	XML	Text including XML markup commands
Image	GIF	Still image in GIF format
	JPEG	Still image in JPEG format
Audio	Basic	Audio, 8-bit PCM sampled at 8000 Hz
	Tone	A specific audible tone
Video	MPEG	Movie in MPEG format
	Pointer	Representation of a pointer device for presentations
Application	Octet-stream	An uninterrupted byte sequence
	Postscript	A printable document in Postscript
	PDF	A printable document in PDF
Multipart	Mixed	Independent parts in the specified order
	Parallel	Parts must be viewed simultaneously

# Architectural Overview



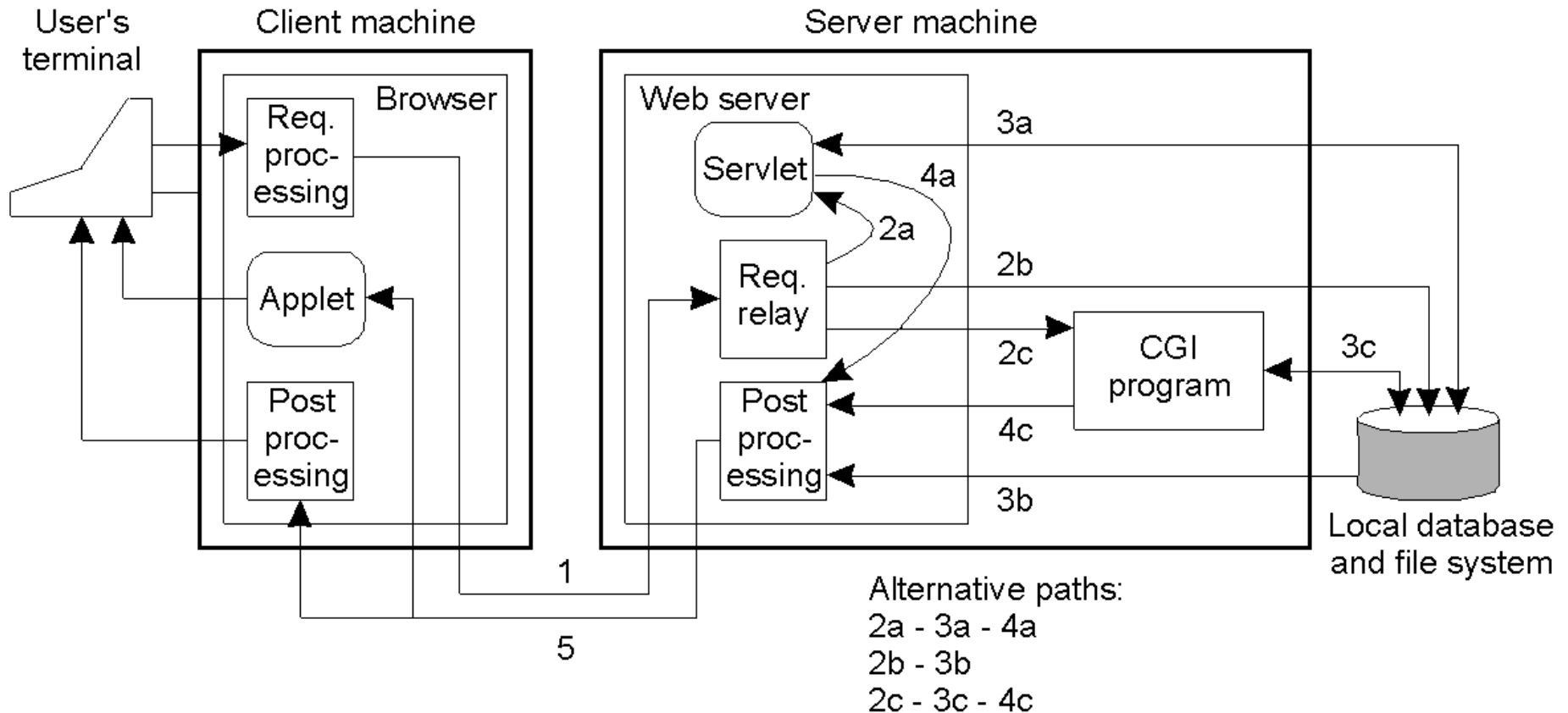
- Using Common Gateway Interface (CGI) to manipulate a database local to the webserver.

# Architectural Overview (2)

```
(1) <HTML>
(2) <BODY>
(3) <P>The current content of <pre>/data/file.txt</PRE>is:</P>
(4) <P>
(5) <SERVER type = "text/javascript");
(6)     clientFile = new File("/data/file.txt");
(7)     if(clientFile.open("r")){
(8)         while (!clientFile.eof())
(9)             document.writeln(clientFile.readln());
(10)        clientFile.close();
(11)    }
(12) </SERVER>
(13) </P>
(14) <P>Thank you for visiting this site.</P>
(15) </BODY>
(16) </HTML>
```

- Javascript to be executed by the server.

# Architectural Overview (3)

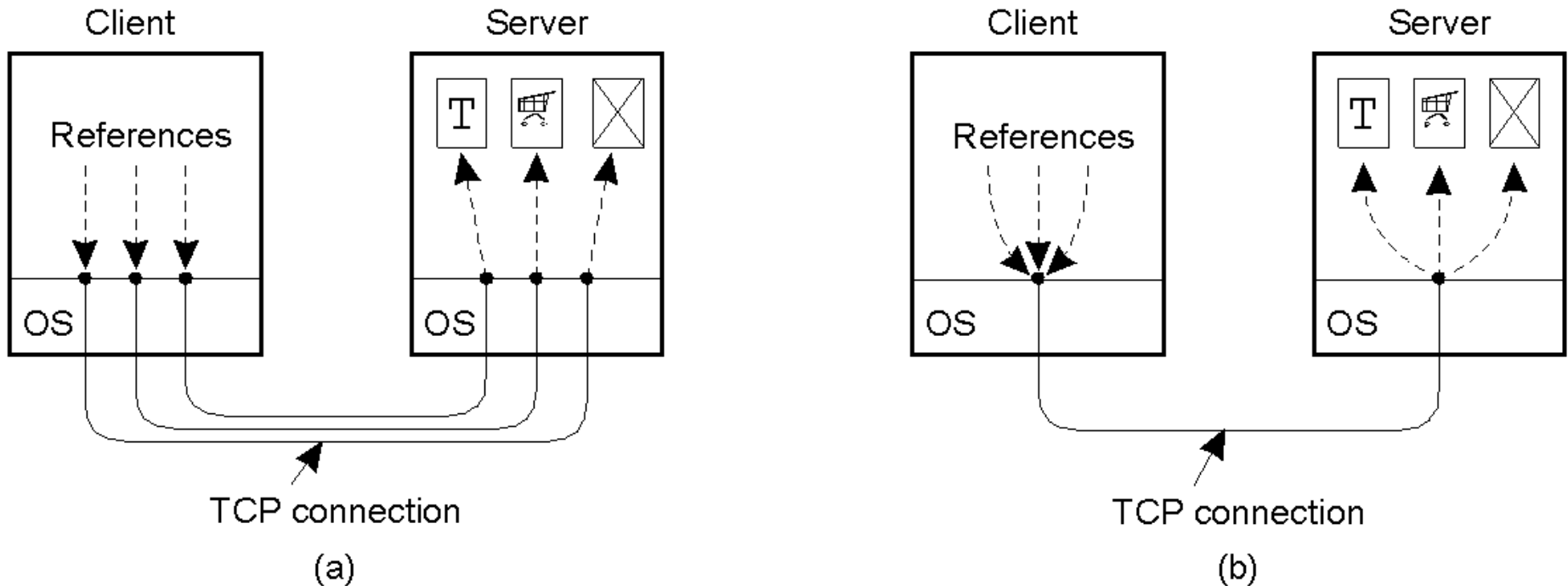


- More Complete Overview.
  - a) Pass Request to a Servlet.
  - b) Retrieve document from database.
  - c) Start CGI program to create document.

# WWW

- Communication
  - Using the Hypertext Transfer Protocol (HTTP)
  - HTTP is stateless and based on TCP.
  - A client sets up a TCP connection to the server and sends a request message, receiving the response message through the same connection.
  - HTTP need not be concerned about lost requests and responses – clients and servers assume their message is received with no attempt made to recover from time out errors.

# HTTP Connections

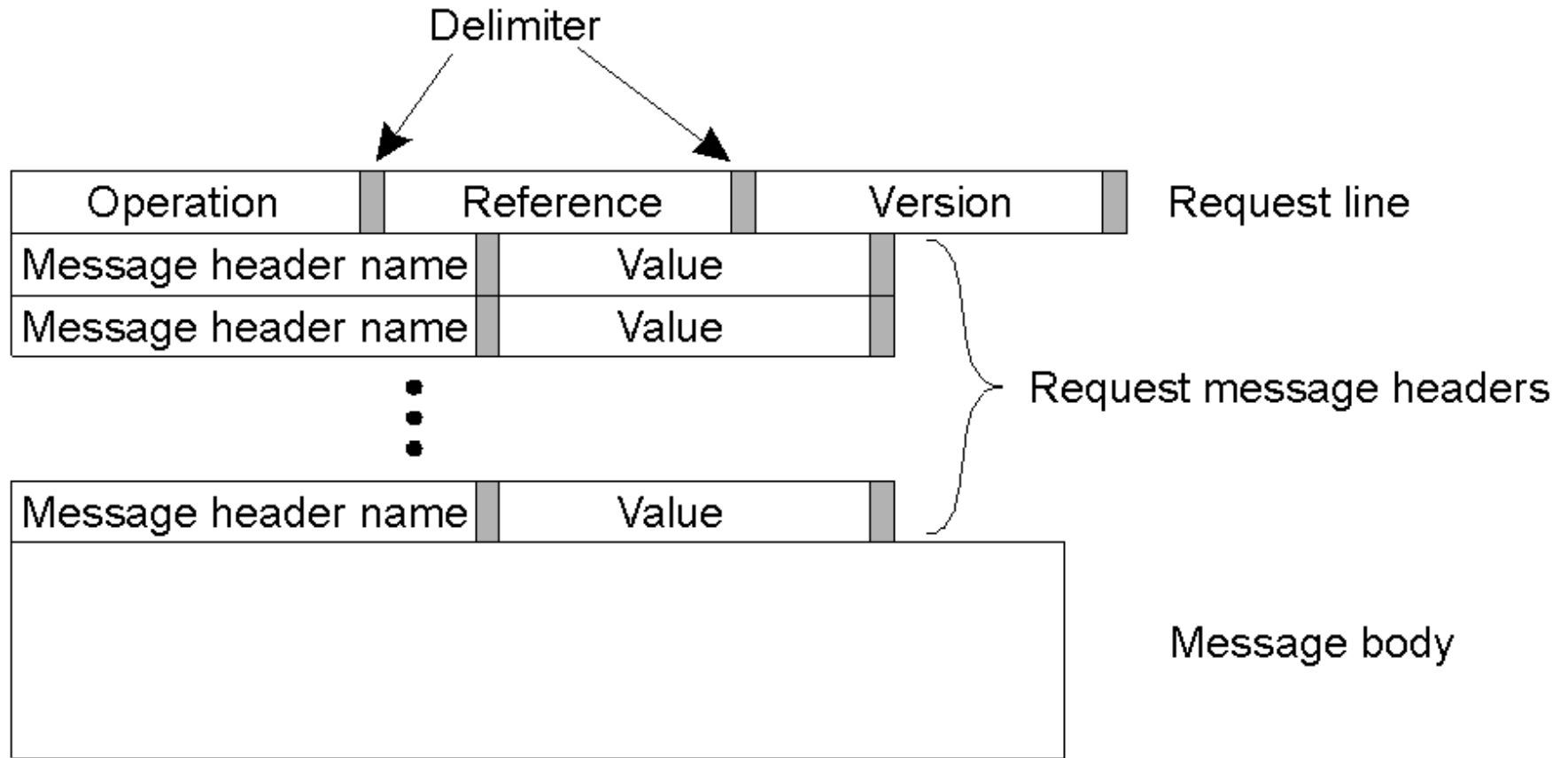


- a) Using nonpersistent connections (HTTP 1.0)
- b) Using persistent connections (HTTP 1.1+)

# HTTP Methods

<b>Operation</b>	<b>Description</b>
Head	Request to return the header of a document
Get	Request to return a document to the client
Put	Request to store a document
Post	Provide data that is to be added to a document (collection)
Delete	Request to delete a document

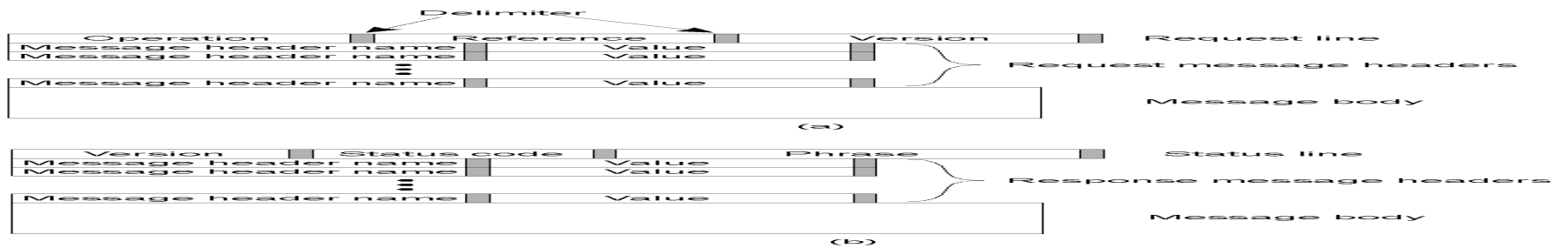
# HTTP Messages



(a)

- A HTTP Request Message

# HTTP Messages(2)



- HTTP Response message

# Status Line

- The status line in a response message has a code and a textual phrase such as;
  - 200 (OK)
  - 400 (Bad Request)
  - 403 (Forbidden)
  - 404 (Not Found)
  - 405 (Method Not Allowed)

# Message Headers

Header	Source	Contents
Accept	Client	The type of documents the client can handle
Accept-Charset	Client	The character sets are acceptable for the client
Accept-Encoding	Client	The document encodings the client can handle
Accept-Language	Client	The natural language the client can handle
Authorization	Client	A list of the client's credentials
WWW-Authenticate	Server	Security challenge the client should respond to
Date	Both	Date and time the message was sent
ETag	Server	The tags associated with the returned document
Expires	Server	The time how long the response remains valid
From	Client	The client's e-mail address
Host	Client	The TCP address of the document's server
If-Match	Client	The tags the document should have
If-None-Match	Client	The tags the document should not have
If-Modified-Since	Client	Tells the server to return a document only if it has been modified since the specified time
If-Unmodified-Since	Client	Tells the server to return a document only if it has not been modified since the specified time
Last-Modified	Server	The time the returned document was last modified
Location	Server	A document reference to which the client should redirect its request
Referer	Client	Refers to client's most recently requested document
Upgrade	Both	The application protocol the sender wants to switch to
Warning	Both	Information about the status of the data in the message

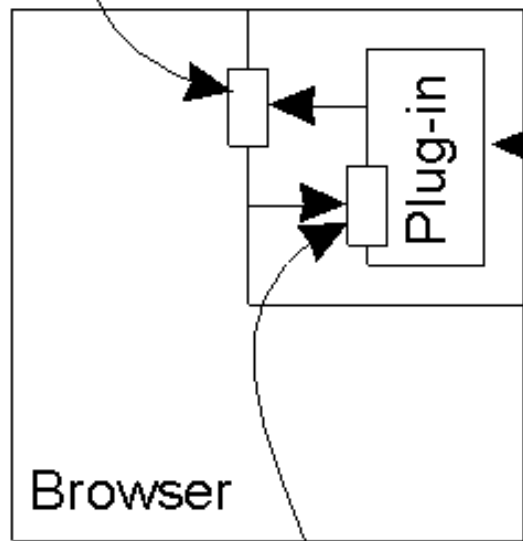
# WWW

- Processes
  - 2 types, browsers for clients and webserver to respond to browser requests.
  - Browsers may be supported by helper programs, and web servers may be supported by additional programs such as CGI scripts.

# Clients

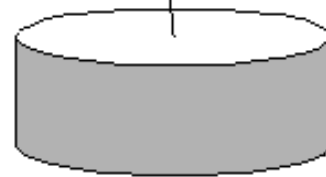
- Clients use browsers, which fetch documents and display them on the user's screen. As they need to handle a variety of document types they can be complex.

Browser's interface for plug-in



Plug-in's interface for browser

Plug-in is loaded on demand



Local file system

A plug is a small program that can be dynamically loaded into a browser to display a specific document type.

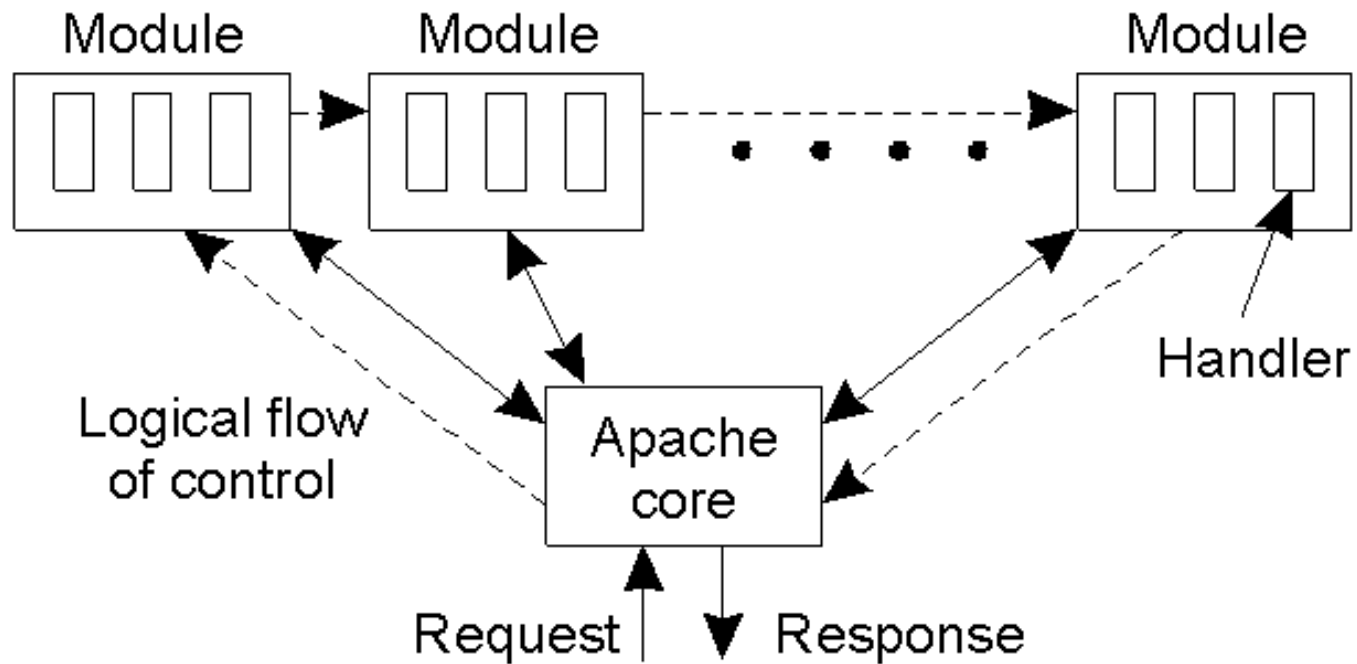
# Proxies

- 
- Another client side process is a web proxy. Originally to enable browsers to handle other protocols (such as FTP above).
  - Now most browsers can deal with most protocols, but are used for caching data to be shared between browsers.

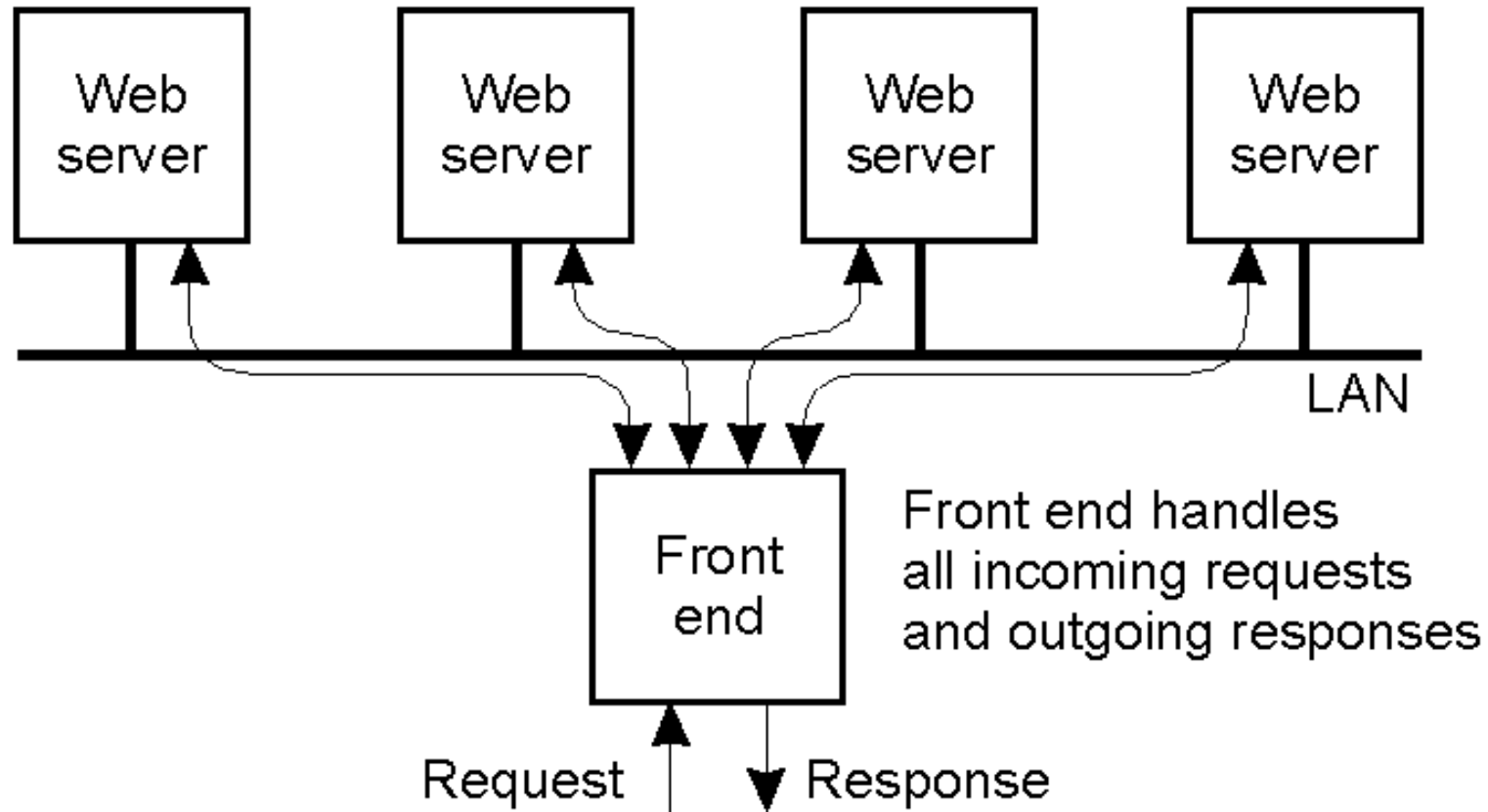
# Webserver Processes

- An Apache Module;
  - 1) Resolving the document reference to a local file name.
  - 2) Client Authentication
  - 3) Client Access Control
  - 4) Request Access Control
  - 5) MIME type determination of the response
  - 6) General Phase for handling leftovers
  - 7) Transmission of the response
  - 8) Logging data on the processing of the request
- At each phase the module decides whether to handle the request, decline it or report an error.

# General Organisation

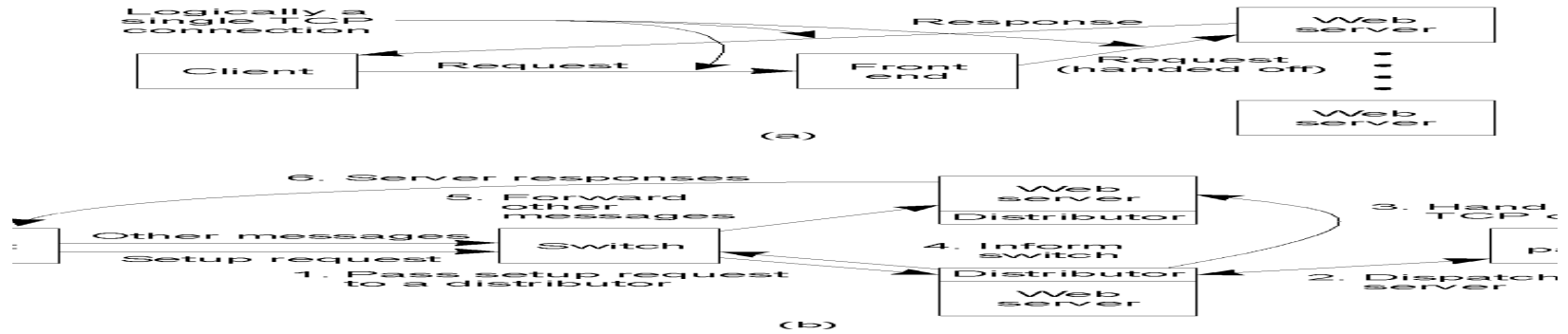


# Server Clusters



- Improving efficiency by clustering servers. But, 'Front end' has a lot of work.

# TCP hand off



After being assigned the webserver replies directly.

# WWW

- Naming
  - Through Uniform Resource Identifiers (URI), Uniform Resource Locators (URL) and Uniform Resource Names (URN)

# URLs

Scheme	Host name	Pathname
http	:// www.cs.vu.nl	/home/steen/mbox

(a)

Scheme	Host name	Port	Pathname
http	:// www.cs.vu.nl	: 80	/home/steen/mbox

(b)

Scheme	Host name	Port	Pathname
http	:// 130.37.24.11	: 80	/home/steen/mbox

(c)

- Often-used structures for URLs.
- b) Using only a DNS name.
- c) Combining a DNS name with a port number.
- d) combining an IP address with a port number.

# URLs

Name	Used for	Example
http	HTTP	http://www.cs.vu.nl:80/globe
ftp	FTP	ftp://ftp.cs.vu.nl/pup/minx/README
file	Local file	file:/edu/book/work/chp/11/11
data	Inline data	data:text/plain;charset=iso-8859-7,%e1%e2%e3
telnet	Remote login	telnet://flits.cs.vu.nl
tel	Telephone	tel:+31201234567
modem	Modem	modem:+31201234567;type=v32

URLs provide easy resolution using a DNS

# URNs

- Structure of a URN.

URN: Namespace: Name of resource

- A URN is a location independent reference to a document. While defining a URN is easy, resolving one is not.

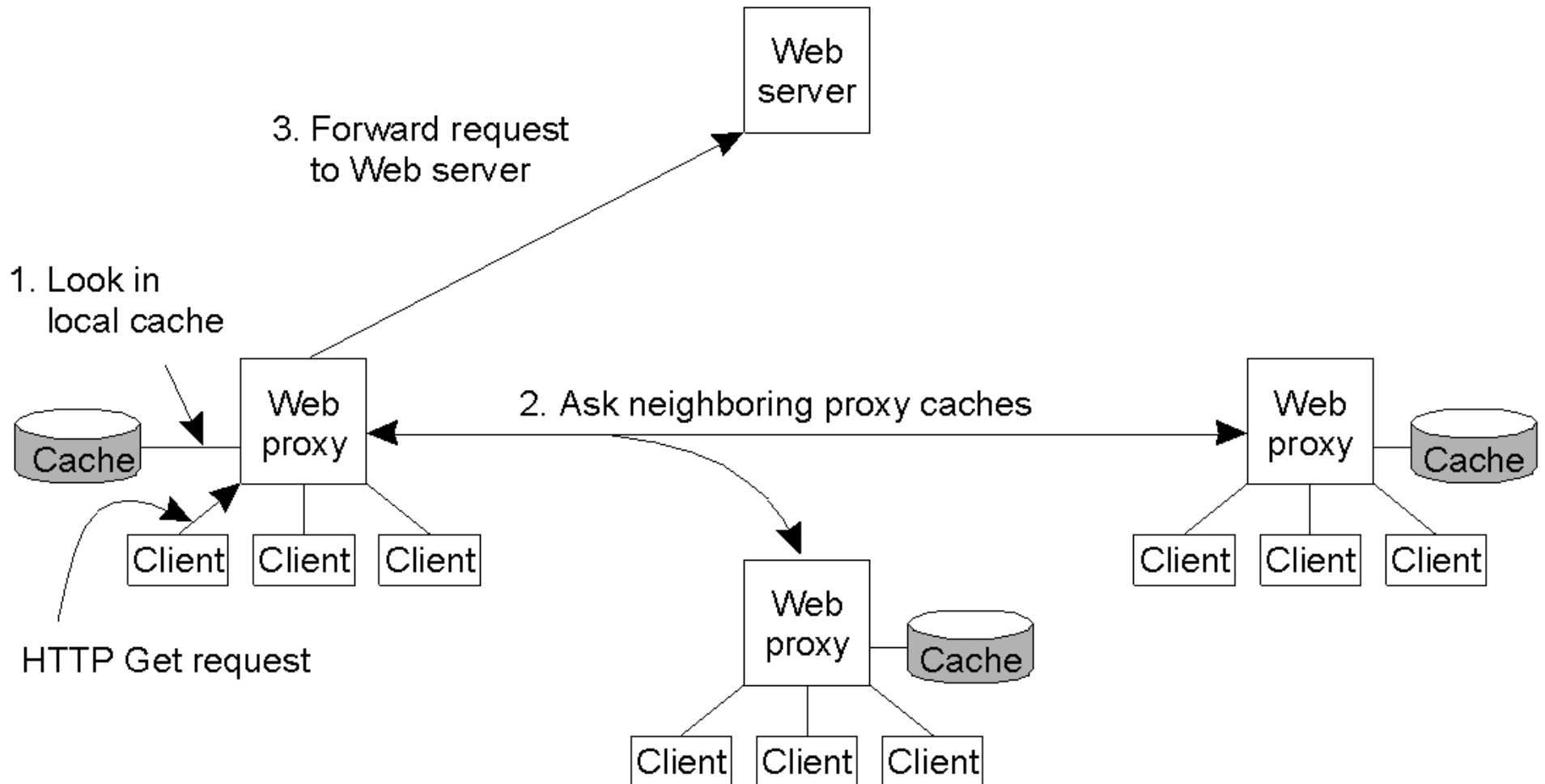
# WWW

- Synchronisation
  - Not really an issue!
    - Strict Client Server relationship means that servers don't need to communicate with other servers.
    - A 'read mostly' environment where write conflicts are unlikely.

# WWW

- Consistency and Replication
  - Web proxy Caching
  - or alternatively Server Replication

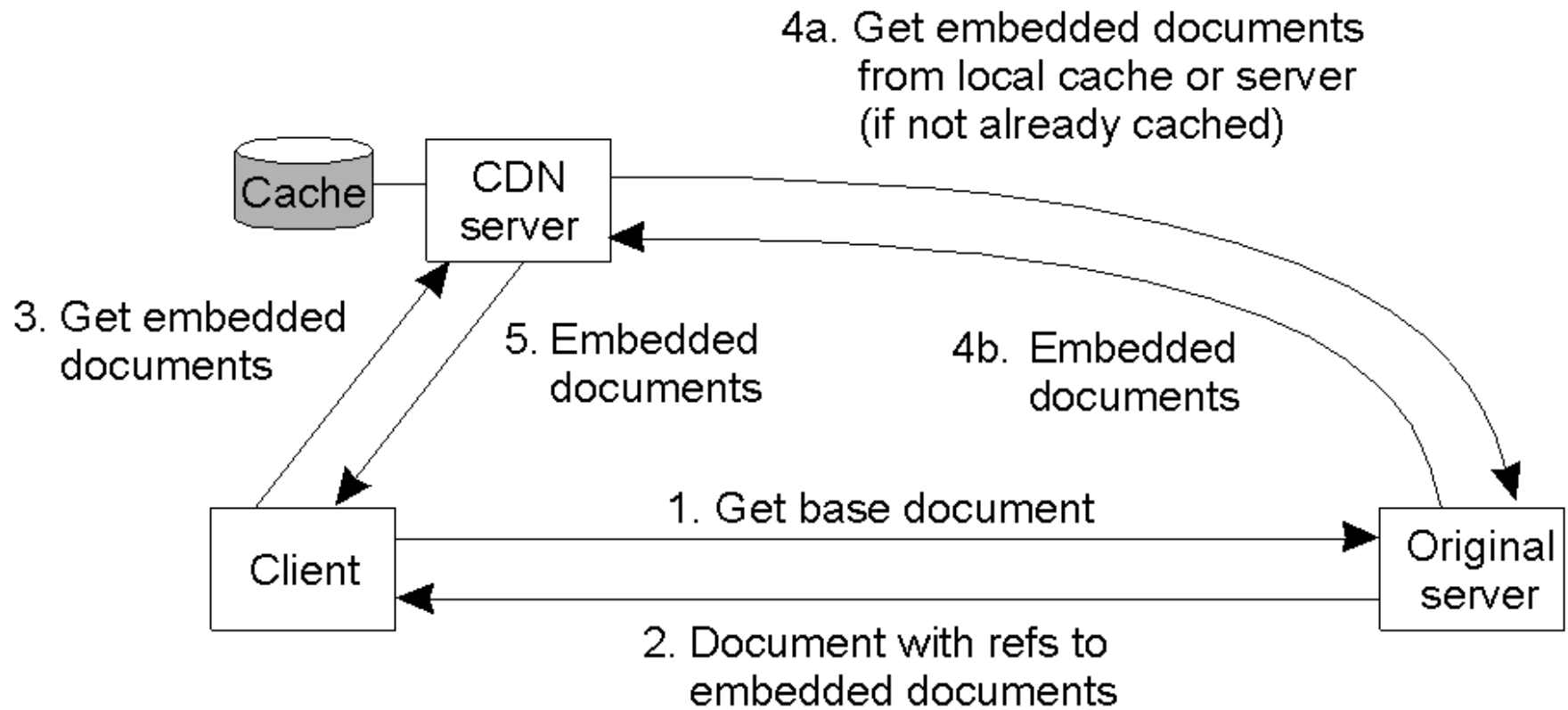
# Web Proxy Caching



# Server Replication

- Use of Web clusters to improve performance on heavily loaded web sites (as already discussed)
- Mirroring, where an entire copy of the site is made available at a different server.
- Using a Content Delivery Network (CDN) to differentiate between the content and the base document.

# CDN



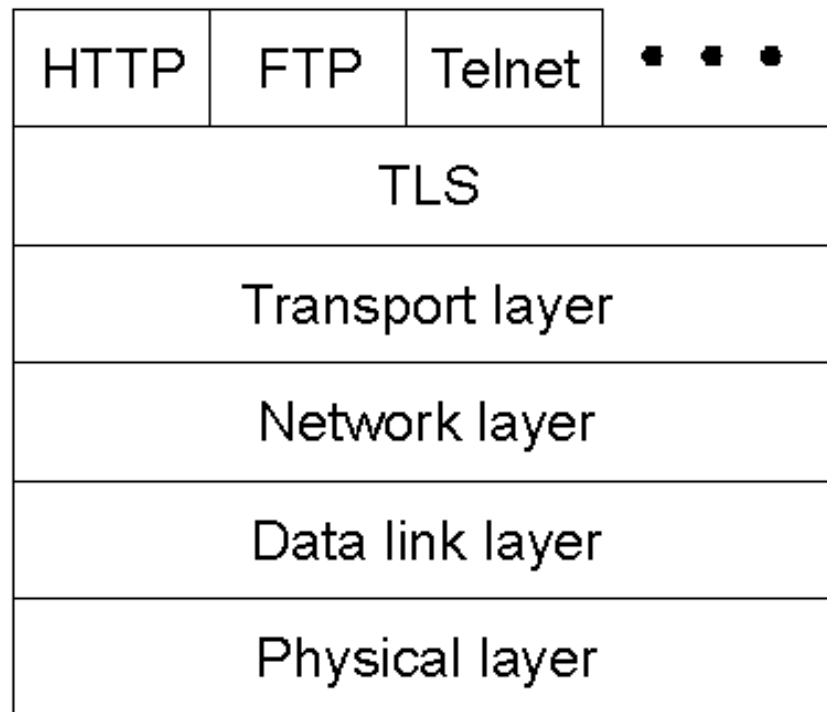
# WWW

- Fault Tolerance
  - Redundancy
  - Time outs

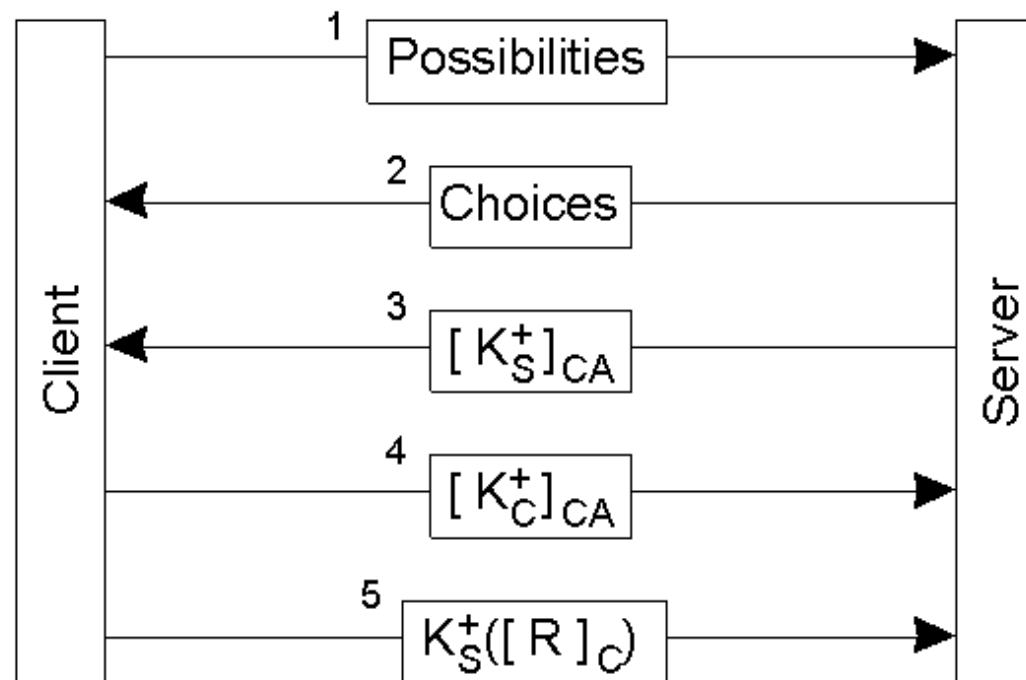
# WWW

- Security
  - SSL (Secure Socket Layer)
  - and more recently TLS (Transport Layer Security)

# TLS



# TLS Authentication



# Other Document Based DS

- Can you think of any other examples of document based distributed systems?

