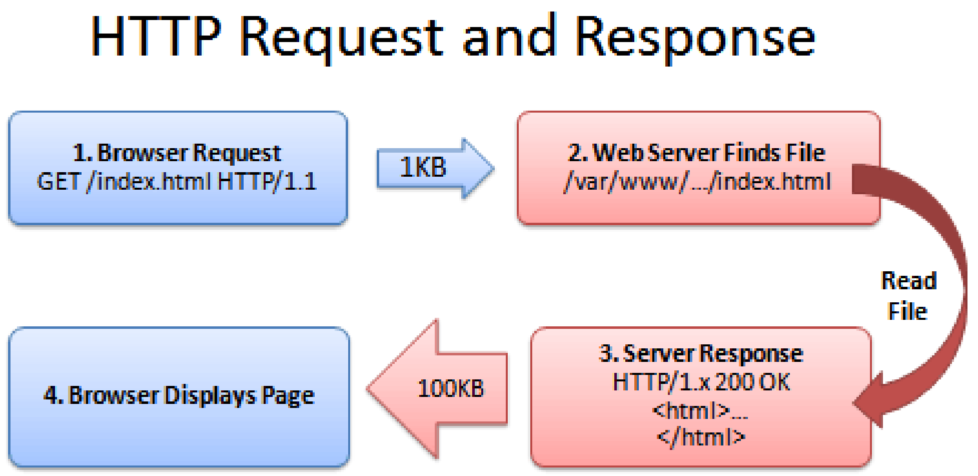
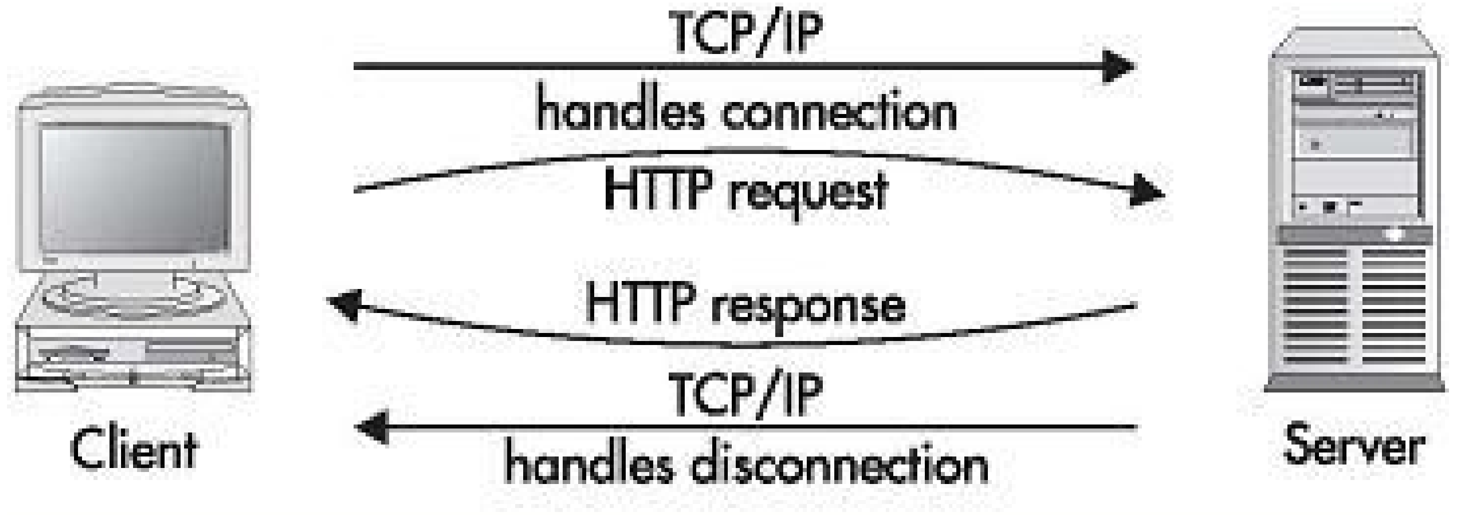
Part A: Introduction to HTTP

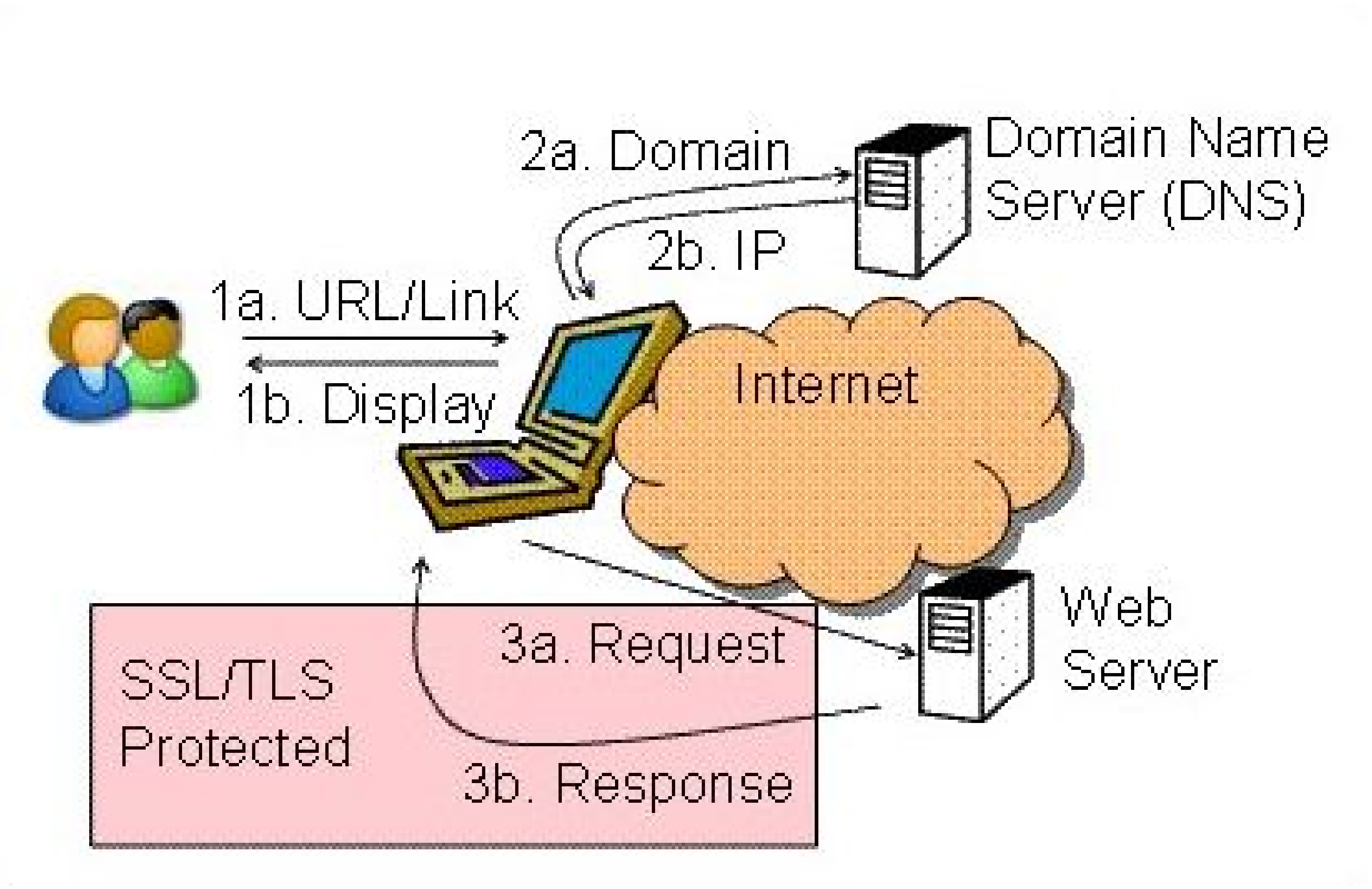
I. Hypertext Transfer Protocol (HTTP)

- It is the network protocol used to deliver resources (Web documents and other data) on the World Wide Web based on TCP/IP.
- It is client-server model. The default port for HTTP servers to listen on is 80.
- HTTP is used to transmit *resources* on the Internet.
  - A resource is some chunk of information that can be identified by a URL.
  - The most common kind of resource is a file, but a resource may also be a dynamically-generated query result, the output of a CGI script, a document that is available in several languages, or something else.





- Before any communication, a connection has to be established between the client and the server.
- An *HTTP client* opens a connection and sends a *request message* to an *HTTP server*; the server then returns a *response message*.
- After delivering the response, the server closes the connection.


2a. Domain
Domain Name Server (DNS)
2b. IP
1a. URL/Link
1b. Display
Internet
Web Server
3a. Request
SSL/TLS Protected
3b. Response

II. Initial Request Line

- A request line has three parts, separated by spaces: a *method name*, *the local path of the requested resource*, and *the version of HTTP being used*.
- A typical request line is:

  ```
  GET /path/to/file/index.html HTTP/1.0
  ```

  - **GET** (uppercase) is the most common HTTP method; it says "give me this resource". Other methods include **POST** and **HEAD**.
  - The path is the part of the URL after the host name, also called the *request URI* (a URI is like a URL, but more general).
  - The HTTP version always takes the form "**HTTP/x.x**", uppercase.
- Sample request message sent by Internet Explorer (IE) for this web page as follows:

  ```
  GET /simtec/httpgallery/introduction/ HTTP/1.1
  Accept:*/*
  Accept-Language: en-gb
  Accept-Encoding: gzip, deflate
  User-Agent: Mozilla/4.0 (compatible; MSIE 6.0)
  Host: www.httpwatch.com
  Connection: Keep-Alive
  ```

- The first line of the message, known as the request line, contains:
  - The HTTP method.
  - The relative URL of the resource or a full URL if you are using an HTTP proxy
  - The version of HTTP that is being used.
  - The rest of the message consists of a set of name/value pairs, known as headers. HTTP clients use header values to control how the request is processed by the server.

III. Initial Response Line (Status Line)

- The initial response line has three parts separated by spaces: the HTTP version, a *response status code* that gives the result of the request, and an English *reason phrase* describing the status code.
- Typical status lines are:

```
HTTP/1.0 200 OK
```

or

```
HTTP/1.0 404 Not Found
```

- The HTTP version is in the same format as in the request line, "HTTP/x.x".
- The status code is meant to be computer-readable; the reason phrase is meant to be human-readable, and may vary.
- The status code is a three-digit integer, and the first digit identifies the general category of response:

  1xx: indicates an informational message only
  2xx: indicates success of some kind
  3xx: redirects the client to another URL
  4xx: indicates an error on the client's part
  5xx: indicates an error on the server's part

The most common status codes are:

`200 OK` : The request succeeded, and the resulting resource (e.g. file or script output) is returned in the message body.
`404 Not Found` : The requested resource doesn't exist.
`301 Moved Permanently`
`302 Moved Temporarily`
`303 See Other` *(HTTP 1.1 only)*: The resource has moved to another URL (given by the **`Location:`** response header), and should be automatically

retrieved by the client. This is often used by a CGI script to redirect the browser to an existing file.

`500 Server Error`: An unexpected server error. The most common cause is a server-side script that has bad syntax, fails, or otherwise can't run correctly.

IV. The Message Body

- An HTTP message may have a body of data sent after the header lines.
- **In a response**, it is where the requested resource is returned to the client or perhaps explanatory text if there's an error. **In a request**, this is where user-entered data or uploaded files are sent to the server.
- If an HTTP message includes a body, there are usually header lines in the message that describe the body. In particular,
    - The **`Content-Type:`** header gives the MIME-type of the data in the body, such as **`text/html`** or **`image/gif`**.
    - The **`Content-Length:`** header gives the number of bytes in the body.

V. Sample HTTP Request and Response

- To retrieve the file at the URL

```
http://www.somehost.com/path/file.html
```

- First open a socket to the host **www.somehost.com**, port 80. Then, send the following request through the socket:

```
GET /path/file.html HTTP/1.0
From: someuser@jmarshall.com
User-Agent: HTTPTool/1.0
[blank line here]
```

- The server should respond with something like the following, sent back through the same socket:

```
HTTP/1.0 200 OK
Date: Fri, 31 Dec 1999 23:59:59 GMT
Content-Type: text/html
Content-Length: 1354
```

```
<html>
<body>
<h1>Happy New Year!</h1>
(more file contents)
  .
  .
  .
</body>
</html>
```

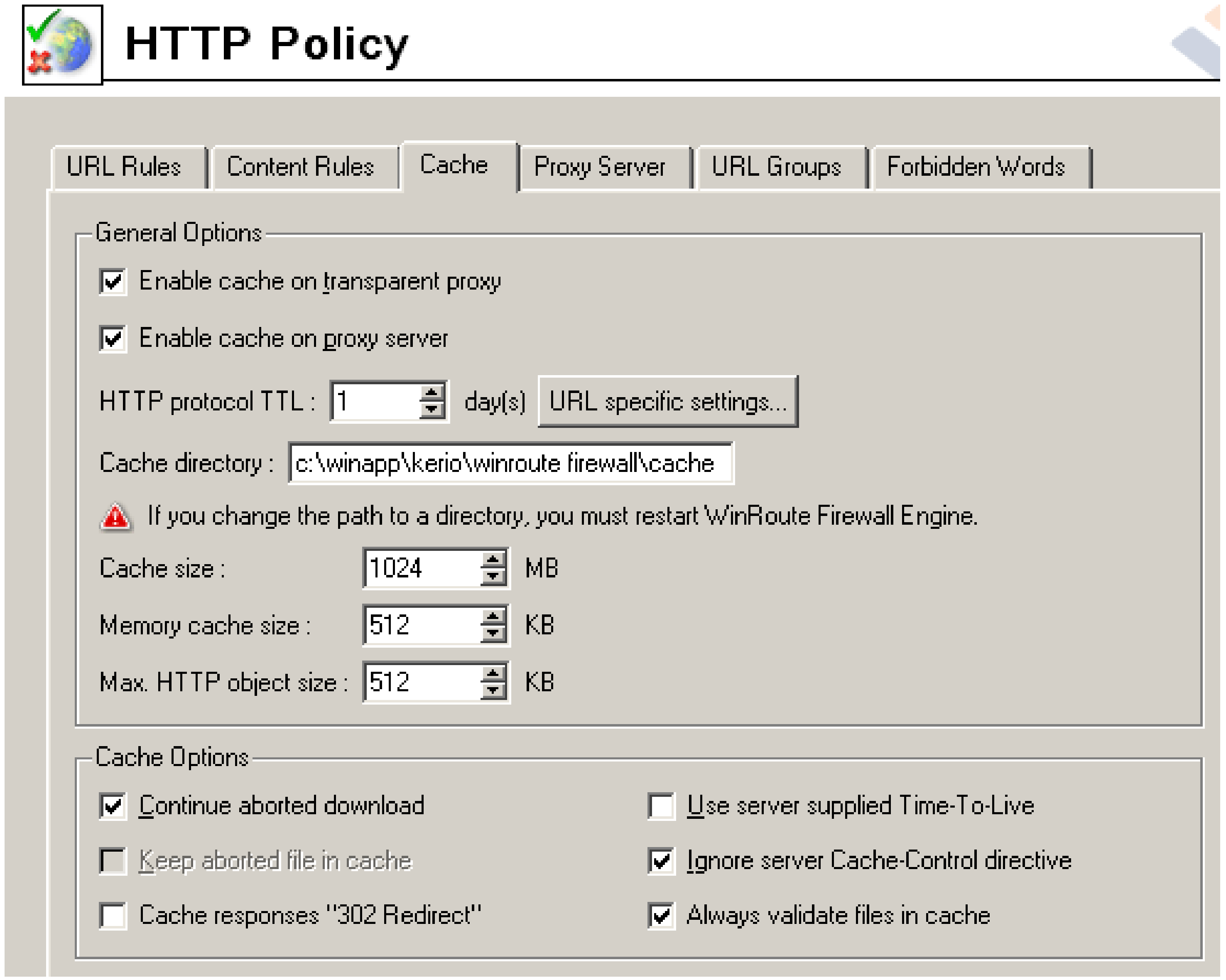
VI. Other HTTP Methods, Like HEAD and POST

- Besides GET, the two most commonly used methods are HEAD and POST.
- The HEAD Method
  - o A HEAD request is just like a GET request, except it asks the server to return the response headers only, and not the actual resource (i.e. no message body).
  - o This is useful to check characteristics of a resource without actually downloading it, thus saving bandwidth.
  - o The response to a HEAD request must *never* contain a message body, just the status line and headers.

- The POST Method
  - A POST request is used to send data to the server to be processed in some way, like by a CGI script. A POST request is different from a GET request in the following ways:
    - There's a block of data sent with the request, in the message body. There are usually extra headers to describe this message body, like “**`Content-Type:`**” and “**`Content-Length:`**”.
    - The *request URI* is not a resource to retrieve; it's usually a program to handle the data you're sending.

VII. Caching: agents between the server and client

- **Browser cache**
  - The browser stored a local copy of the visited Web pages when they were first downloaded. These local copies are kept in what's called a **cache**.
  - Usually one sets a maximum size for the cache and a maximum caching time for documents.
  - **Caching Algorithm**: When a new page is visited it is stored in the cache and if the cache is full, some document that the browser considers unlikely to be visited again soon is deleted to make room.



HTTP Policy

General Options

- [✓] Enable cache on transparent proxy
- [✓] Enable cache on proxy server

HTTP protocol TTL : 1 day(s) URL specific settings...

Cache directory : c:\winapp\kerio\winroute firewall\cache

If you change the path to a directory, you must restart WinRoute Firewall Engine.

Cache size : 1024 MB

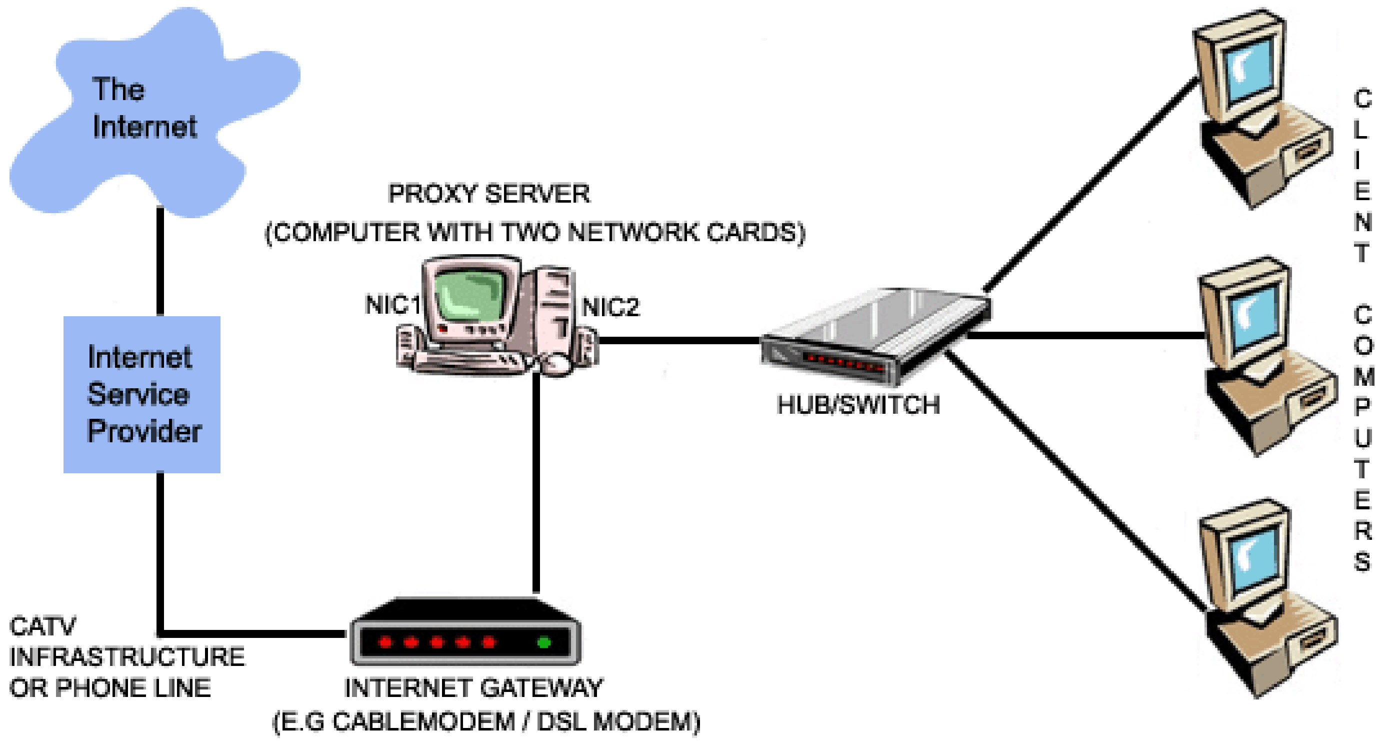
Memory cache size : 512 KB

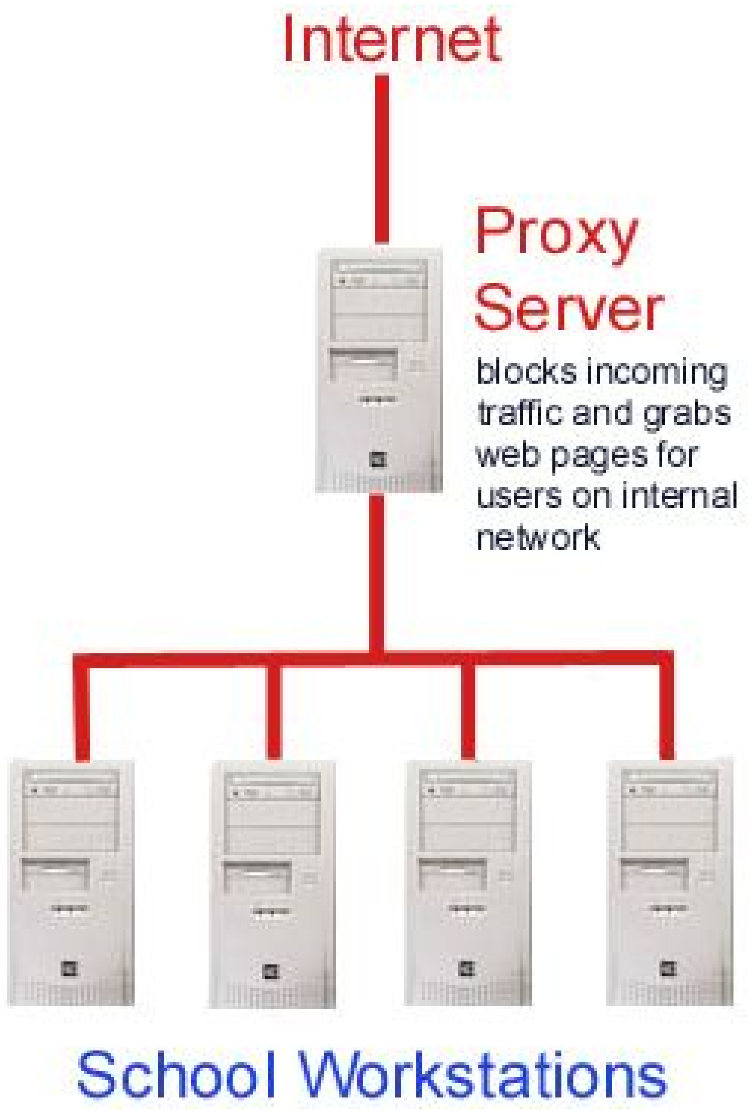
Max. HTTP object size : 512 KB

Cache Options

- [✓] Continue aborted download
- [ ] Keep aborted file in cache
- [ ] Cache responses "302 Redirect"
- [ ] Use server supplied Time-To-Live
- [✓] Ignore server Cache-Control directive
- [✓] Always validate files in cache

- **Proxy caches**
  - For browser caching, when many users browse from the same site one usually ends up storing the same document in many different caches and refreshing it over and over for different uses.
  - To improve the efficiency, all the users should share a common cache.
    - Browsers still have their local caches. If the requested documents are not in the browser cache, the requests will be forwarded to the proxy cache before going to the source Web server.
  - So the proxy can reduce network traffic significantly.
  - A proxy server can be constructed as a hierarchical way. For example, an ISP may have one proxy cache for each part of the country and set up each of the regional proxies to use a national proxy cache.


The Internet
Internet Service Provider
PROXY SERVER
(COMPUTER WITH TWO NETWORK CARDS)
NIC1
NIC2
HUB/SWITCH
CLIENT COMPUTERS
CATV INFRASTRUCTURE OR PHONE LINE
INTERNET GATEWAY
(E.G CABLEMODEM / DSL MODEM)


Internet
Proxy Server
blocks incoming traffic and grabs web pages for users on internal network
School Workstations

VIII. Server-side Programming (Dynamic Web Documents)

- **Background**
  - Server-side scripts or programs are simply programs that are run on the web server in response to requests from the clients.
  - These scripts produce normal HTML as output which is then fed back to the client.
  - There is no way for the client software to tell whether scripting has been used or not.
  - There is a major difference between server-side and client-side scripting as the client and server are usually different computers.
    - If all the data (the program needs) are located on the server it should use server-side scripting instead of client-side.
    - If the program and user need to interact often client-side scripting is probably best, to reduce the number of requests sent to the server.

- **Interactions between client and server**
  - The web server receives a request just like any other, this URL maps to a script file.
  - The server then starts the script, feeding it all the information contained in the request headers and URL.
  - The script then runs and produces as its output the HTML and HTTP headers to be returned to the client, which the server takes care of.
  - CGI (Common Gateway Interface) is a way for web servers and server-side programs to interact.
    - CGI is completely independent of programming language, operating system and web server.
    - Currently it is the most common server-side programming technique and it's also supported by almost every web server in existence.

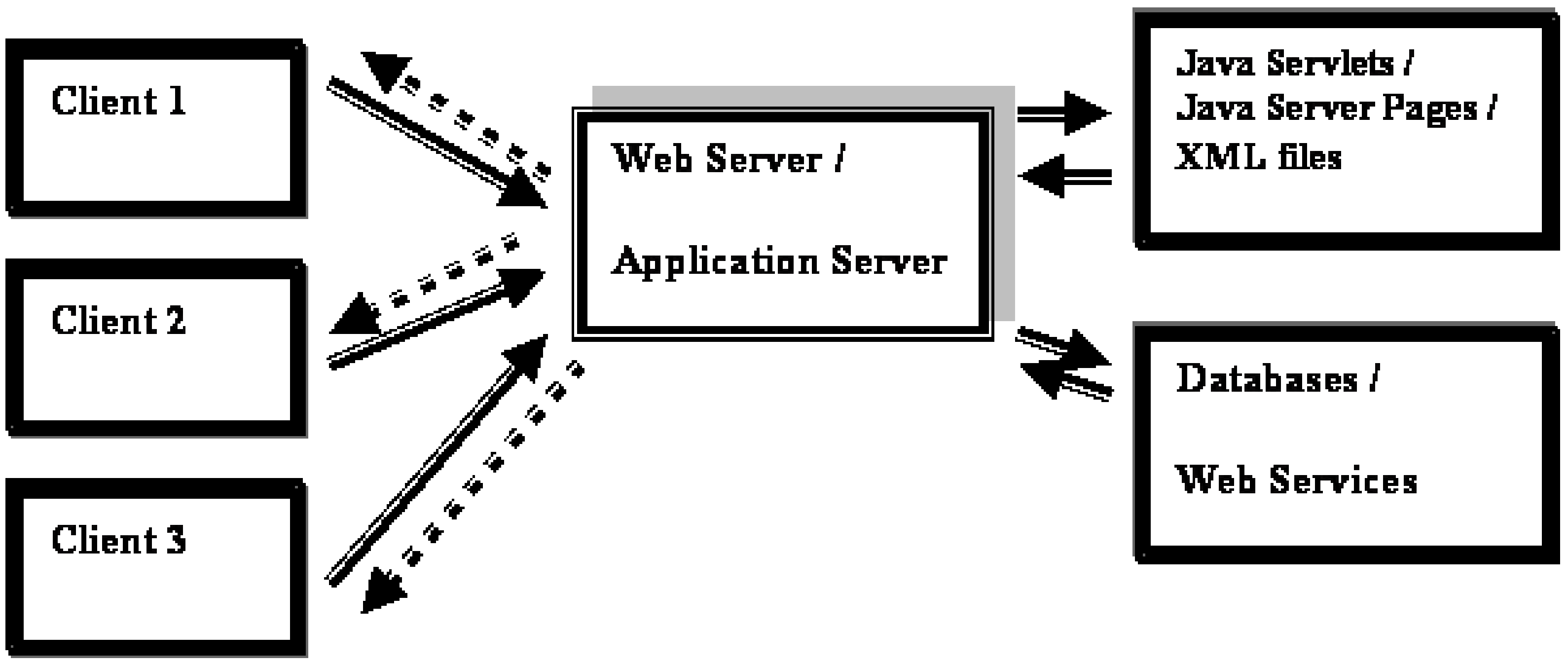
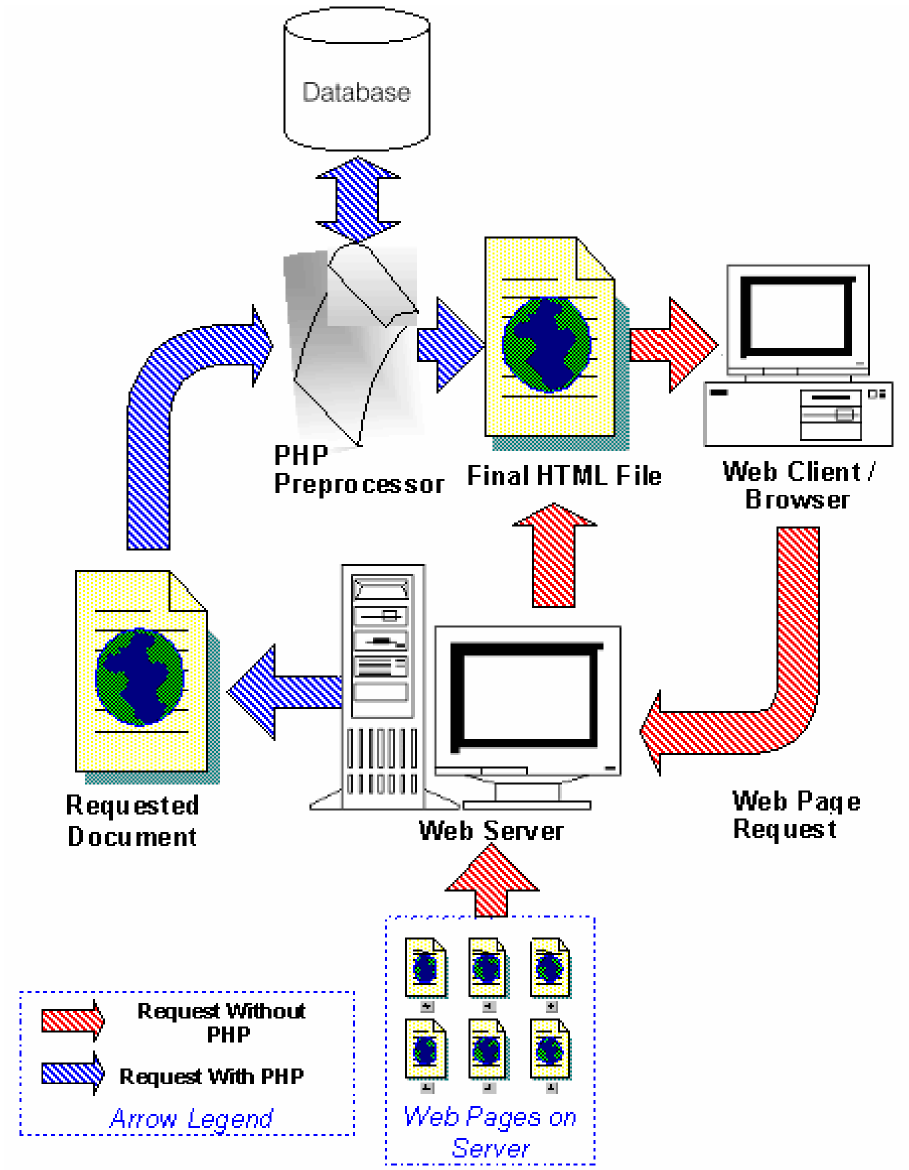


Figure – Server Side Programming (architecture)


Database
PHP Preprocessor
Final HTML File
Web Client / Browser
Requested Document
Web Server
Web Page Request
Request Without PHP
Request With PHP
Arrow Legend
Web Pages on Server