

ମୃଗୟା

# Chapter 1

- **HTTP – the Hyper-Text Transfer Protocol**
- Introduction
- The World Wide Web (WWW)
- The Client/Server Architecture of the WWW
- The Domain Name System (DNS)
- URI, URL, URN
- HTTP Header
- HTTP Request/Response, the Stateless nature of HTTP
- Web browser configuration
- HTTP Authentication

# Introduction

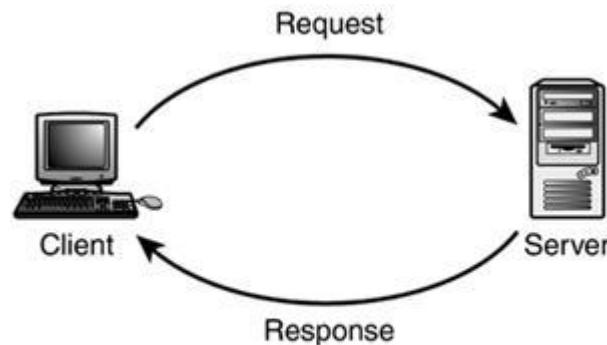
- What is **Internet**?
  - is a big TCP/IP network connecting computers all over the world.
- World Wide Web is an architectural framework for accessing linked documents spread out over millions of machines all over the Internet.
- **The Web** is a collection of Internet resources that use the HTTP protocol.
- A **Web server** is a server that communicates using Hypertext Transfer Protocol (HTTP).

## Cont'd . . . Introduction

- The principal activities on the Internet include :
  - The World Wide Web
  - Email
  - FTP
  - Newsgroups
  - Chat and instant messaging
  - Remote access

# Cont'd . . . Introduction

- Computers on the Internet can be classified as either
  - Clients: computers that request services
  - Servers: computers that provide services



# Client side

- When a user clicks on a hyperlink, the browser carries out a series of steps in order to fetch the page pointed to

<http://www.itu.org/home/index.html>.

- ➔ The browser determines the Host
- ➔ The browser asks DNS for the IP address of [www.itu.org](http://www.itu.org).
- ➔ DNS replies with 156.106.192.32.
- ➔ The browser makes a TCP connection to port 80 on 156.106.192.32.
- ➔ It then sends over a request asking for file /home/index.html.
- ➔ The [www.itu.org](http://www.itu.org) server sends the file /home/index.html.
- ➔ The TCP connection is released.
- ➔ The browser displays all the text in /home/index.html.
- ➔ The browser fetches and displays all images in this file.

# The Server Side

- the steps that the server performs in its main loop are:
  - Accept a TCP connection from a client (a browser).
  - Get and resolve the name of the file requested.
  - Authenticate the client.
  - Perform access control on the client.
  - Perform access control on the Web page.
  - Check the cache.
  - Fetch the requested page from disk.
  - Determine the MIME type to include in the response.
  - Take care of miscellaneous tasks.
  - Return the reply to the client.
  - Make an entry in the server log.
  - Release the TCP connection

# DNS (Domain Name System)

- DNS is the invention of a hierarchical, domain-based naming scheme and using a distributed database system.
- It is primarily used for mapping host names and e-mail destinations to IP addresses

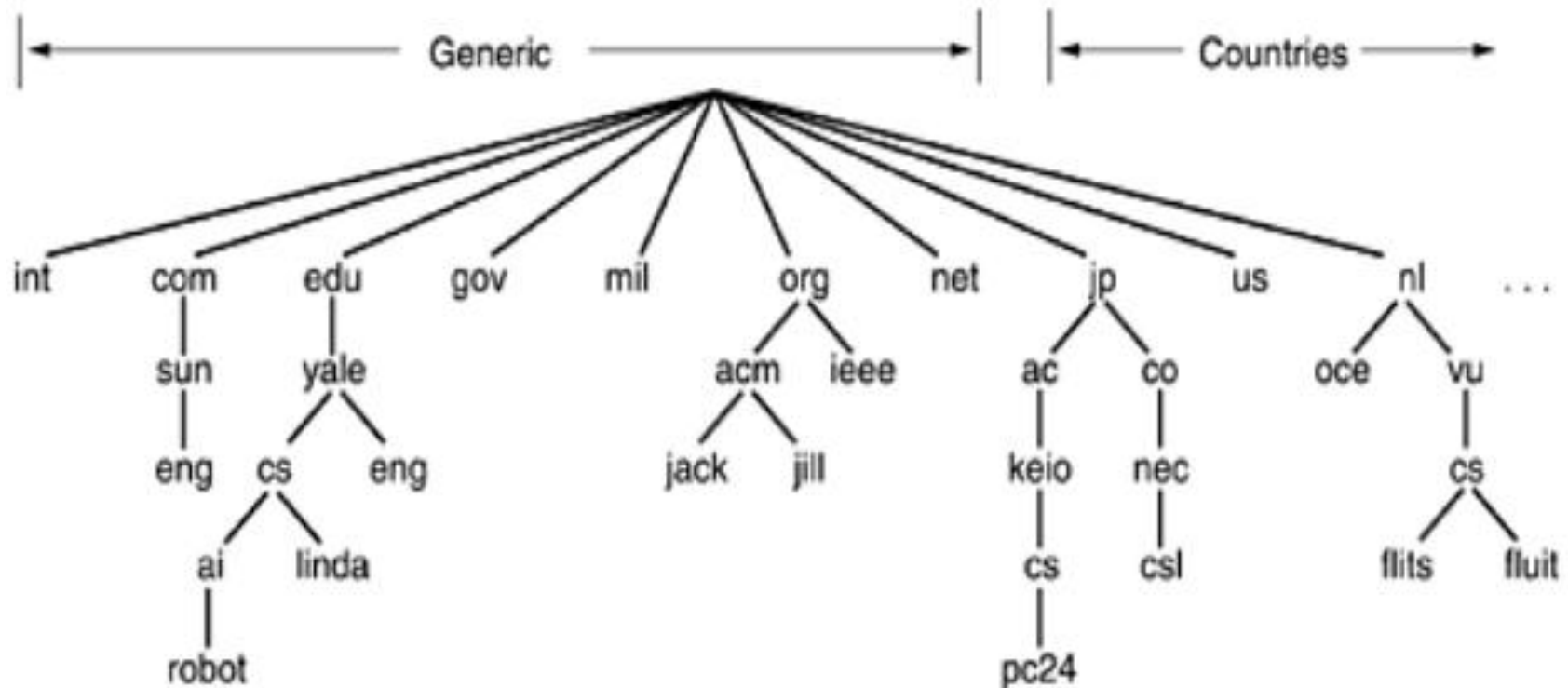
## The DNS Name Space

- The Internet is divide into over 200 top-level domain, which covers many hosts.
- Each domain is partitioned into subdomains,



# Cont'd . . . **DNS (Domain Name System)**

The top-level domains come in two flavors: generic and countries.

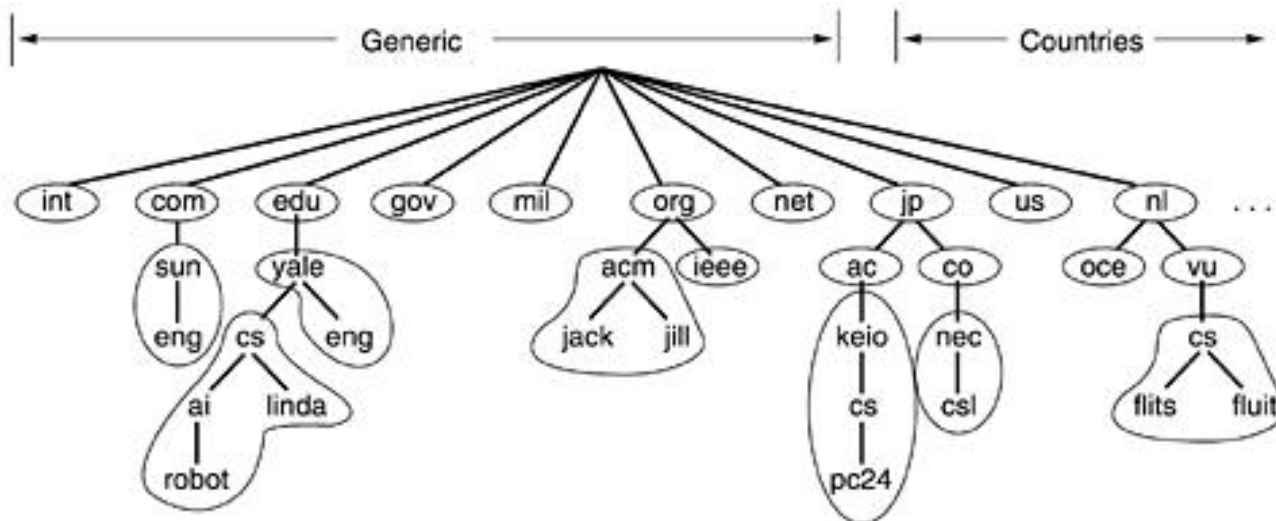


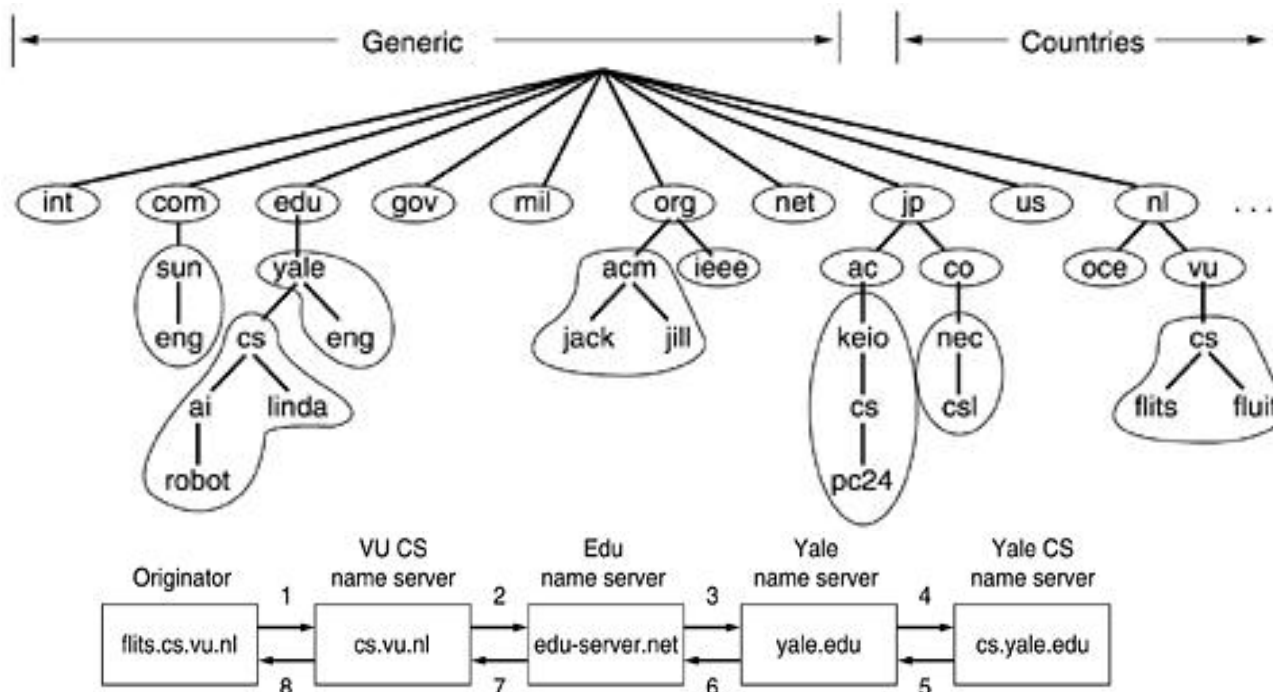
# Cont'd . . . DNS (Domain Name System)

- In general, getting a second-level domain is easy.
  - such as **name-of-company.com**
- every common (English) word has been taken in the **com** domain.
- Domain names are case insensitive
  - **Edu**, **EDU** and **edu** mean the same thing
- Each domain controls how it allocates the domains under it.
  - eg **.et** controls **.gov.et** , **.com.et**
  - **www.aau.edu.et**

# Name Servers

- In theory at least, a single name server could contain the entire DNS database and respond to all queries about it.
- In practice, this server would be so overloaded as to be useless. Furthermore, if it ever went down, the entire Internet would be crippled.
- To avoid the problems associated with having only a single source of information, the DNS name space is divided into nonoverlapping zones.





Eg.

- if a resolver on `flits.cs.vu.nl` wants to know the IP address of the host `linda.cs.yale.edu`. The steps are :
  - it sends a query to the local name server, `cs.vu.nl`.
  - If the sever does not know this domain the request is sent to the top-level domain (`edu-server.net`), even if this server doesnt know the subdomain it must know the `yale.edu`
  - Then from that the subdomains names will be resolved

# URI

- Each web server resource has a name which is called a **uniform resource identifier (URI)**
- **URI** - uniquely identify and locate information resources around the world.
- Eg. URI for an image resource on Joe's Hardware store's web server:
  - <http://www.joes-hardware.com/specials/saw-blade.gif>
- URIs come in two flavors, called
  - URLs and
  - URNs.

# URL (Uniform Resource Locator )

- Three questions had to be answered before a selected page could be displayed:
  - What is the page called?
  - Where is the page located?
  - How can the page be accessed?
- URL will answer all the above questions.

`http://www.abcd.com/products.html`

the name of the  
protocol (http)

the DNS name of  
the machine  
where the page is  
located

the name of the  
file containing  
the page

## Cont'd ...**URL (Uniform Resource Locator )**

- Not all URLs refer to HTTP.
- The general format for a URL is  
<scheme>:<scheme-specific-part>
- The protocol portion of the URL is referred to as the scheme

Some of URL schemes

ftp	File Transfer Protocol
http	Hypertext Transfer Protocol
mailto	Electronic mail
news	Usenet news
telnet	Interactive session
file	Host-specific filenames

# HTTP URL

- The general form for **HTTP URL** is

`http://<host>[:<port>]/<path>[;<parameters>][?<search>]`

- `<host>` - is the DNS name of the server (for example, `www.ambouc.com`), and
- `<path>` - is the path to the HTML document or other resource.
- `<port>` - The port number of service to which the browser is connecting. The port number reserved for HTTP servers is TCP port 80. If the port number is omitted, port 80 is assumed.
- `<parameters>`- Optional parameters supplied by the client.
- `<search>` - Lets the client send a query string to the server through the URL.



# ***URN(uniform resource name)***

- URN serves as a unique name for a particular piece of content, independent of where the resource currently resides.
- these location-independent URNs allow resources to move from place to place.
- URNs are still experimental and not yet widely adopted.
- Eg.

urn:ietf:rfc:2141

might be used to name the Internet standards document "RFC 2141"

# Media Types

- HTTP carefully tags each object being transported through the Web with a data format label called a MIME type.
  - MIME (Multipurpose Internet Mail Extensions)
- MIME was originally for email, worked so well that HTTP adopted it to describe and label its own multimedia content.
- Web servers attach a MIME type to all HTTP object data.
- Most browsers can handle hundreds of popular object types:
  - displaying image files,
  - parsing and formatting HTML files,
  - playing audio files through the computer's speakers, or
  - launching external plug-in software to handle special formats.

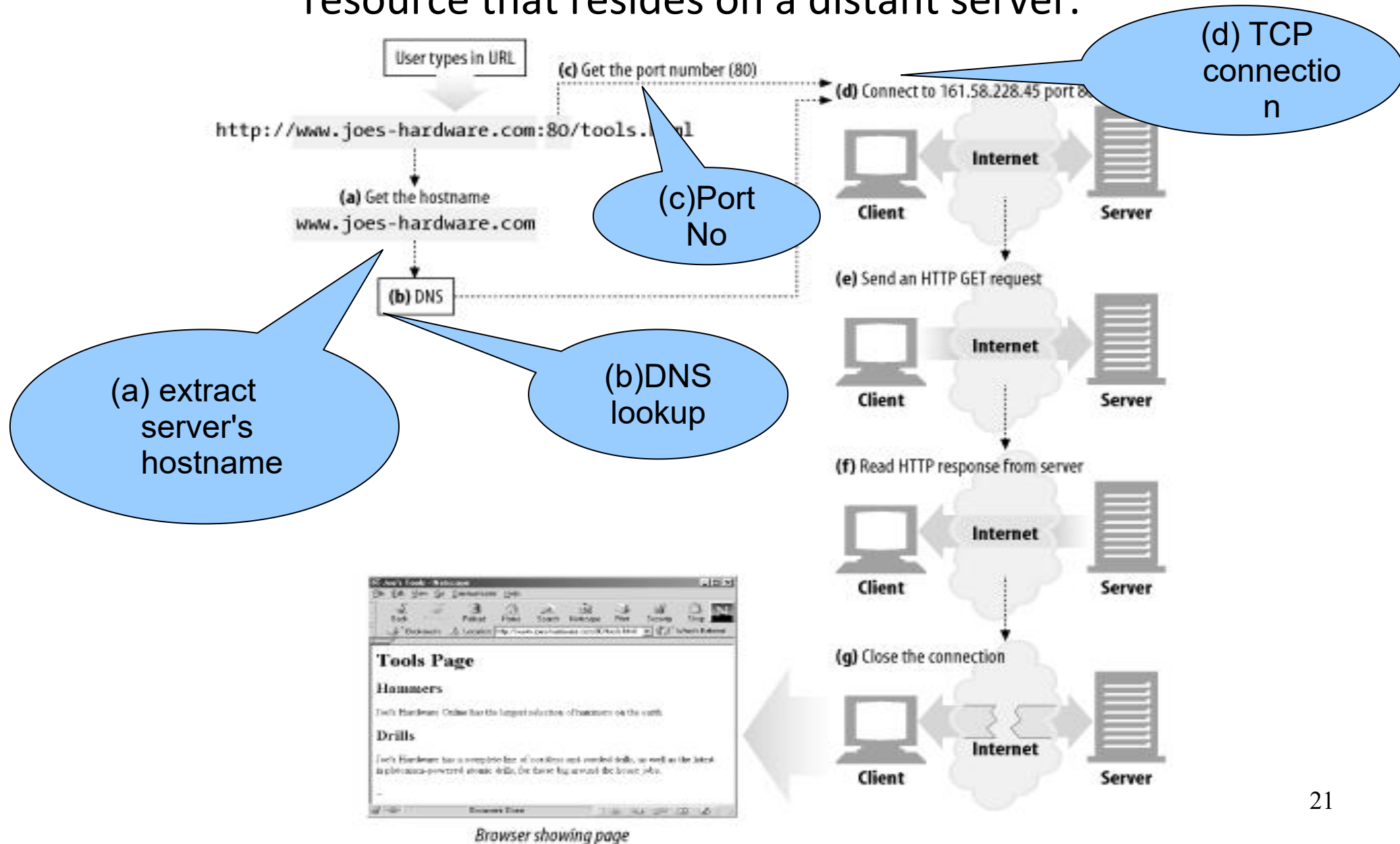
## Cont'd . . . **Media Types**

- A MIME type is a textual label, represented as a primary object type and a specific subtype, separated by a slash. For example:
- An HTML-formatted text document would be labeled with type text/html.
- A plain ASCII text document would be labeled with type text/plain.
- A JPEG version of an image would be image/jpeg.
- A GIF-format image would be image/gif.
- An Apple QuickTime movie would be video/quicktime.
- A Microsoft PowerPoint presentation would be application/vnd.ms-powerpoint.

# HTTP

- is a language between Web servers and browsers
- HTTP has the following duties:
  - To establish a connection between the browser (the client) and the server
  - To negotiate settings and establish parameters for the session
  - To provide for the orderly transfer of HTML content
  - To close the connection with the server

When one enters a URL into the browser window, The Figure below shows how a browser uses HTTP to display a simple HTML resource that resides on a distant server.

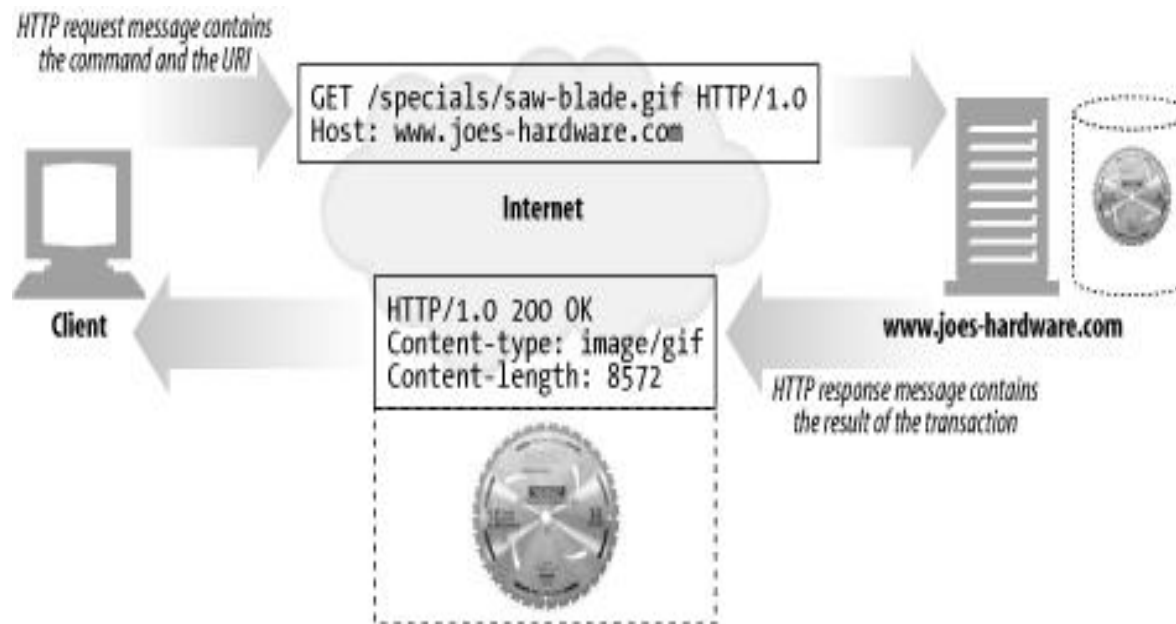


# The steps are:

- (a) The browser extracts the server's hostname from the URL.
- (b) The browser converts the server's hostname into the server's IP address. (The client computer sends the DNS lookup request to a name server and receives the server's IP address. )
- (c) The browser extracts the port number (if any) from the URL. With the IP address and port number, a client can easily communicate via TCP/IP.
- (d) The browser establishes a TCP connection with the web server.
- (e) The browser sends an HTTP request message (HTTP GET command) to the server.
- (f) The server sends an HTTP response back to the browser, along with the document is a header containing several settings.
- (g) The connection is closed, and the browser displays the document

# HTTP Transactions

- HTTP transaction consists of
  - A **request command** (sent from client to server), and
  - a **response result** (sent from the server back to the client).
- This communication happens with formatted blocks of data called **HTTP messages**



# Messages

- HTTP messages are simple, line-oriented sequences of characters.
  - HTTP messages sent from web clients to web servers are called **request messages**.
  - Messages from servers to clients are called **response messages**.

The format for a request message:      The format for a response message

<method> <request-URL> <version>  
<headers>

<version> <status> <reason-phrase>  
<headers>

<entity-body>

<entity-body>

(a) Request message

GET /test/hi-there.txt HTTP/1.0
Accept: text/* Accept-Language: en,fr

*Start line*

*Headers*

*Body*

(b) Response message

HTTP/1.0 200 OK
Content-type: text/plain Content-length: 19
Hi! I'm a message!



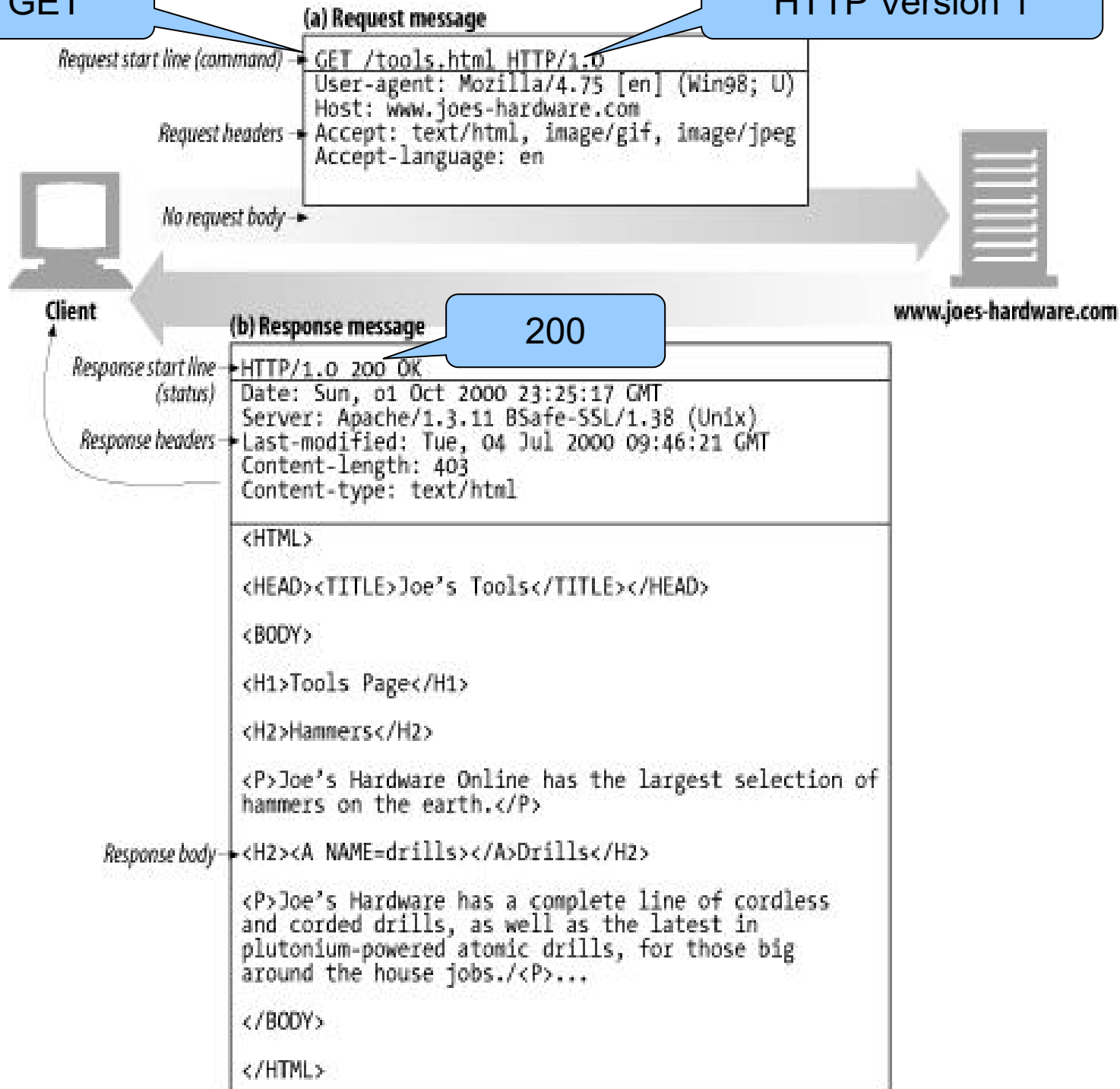
## Cont'd . . . **Messages**

- *Start line*
  - indicates what to do for a request or
  - what happened for a response.
- *Header fields*
  - Each header field consists of a name : value,
- *Body*
  - is an optional message containing any kind of data.
    - Request bodies carry data to the web server;
    - response bodies carry data back to the client.
  - Unlike the start lines and headers, which are textual and structured, the body can contain arbitrary binary data (e.g., images, videos, audio tracks, software applications, text).

Example

GET

HTTP Version 1



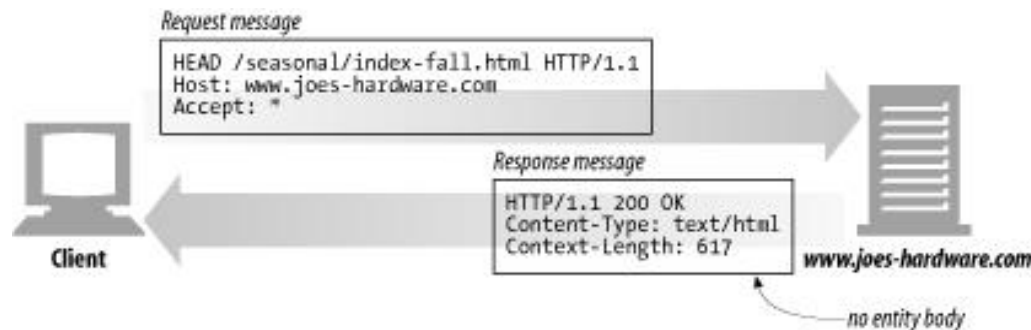
# HTTP methods

- HTTP methods are request commands
- Every HTTP request message has a method. The method tells the server what action to perform

Some common HTTP methods	
HTTP method	Description
GET	Send named resource from the server to the client.
PUT	Store data from client into a named server resource.
DELETE	Delete the named resource from a server.
POST	Send client data into a server gateway application.
HEAD	Send just the HTTP headers from the response for the named resource.

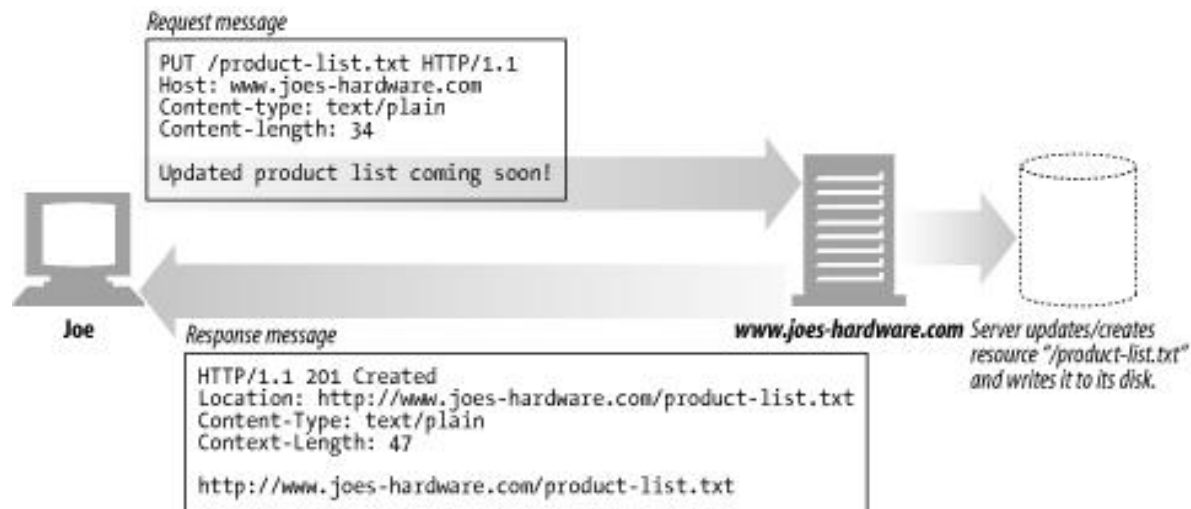
# GET and HEAD

- GET is the most common method. It usually is used to ask a server to send a resource.
- The HEAD method behaves exactly like the GET method, but the server returns only the headers in the response.



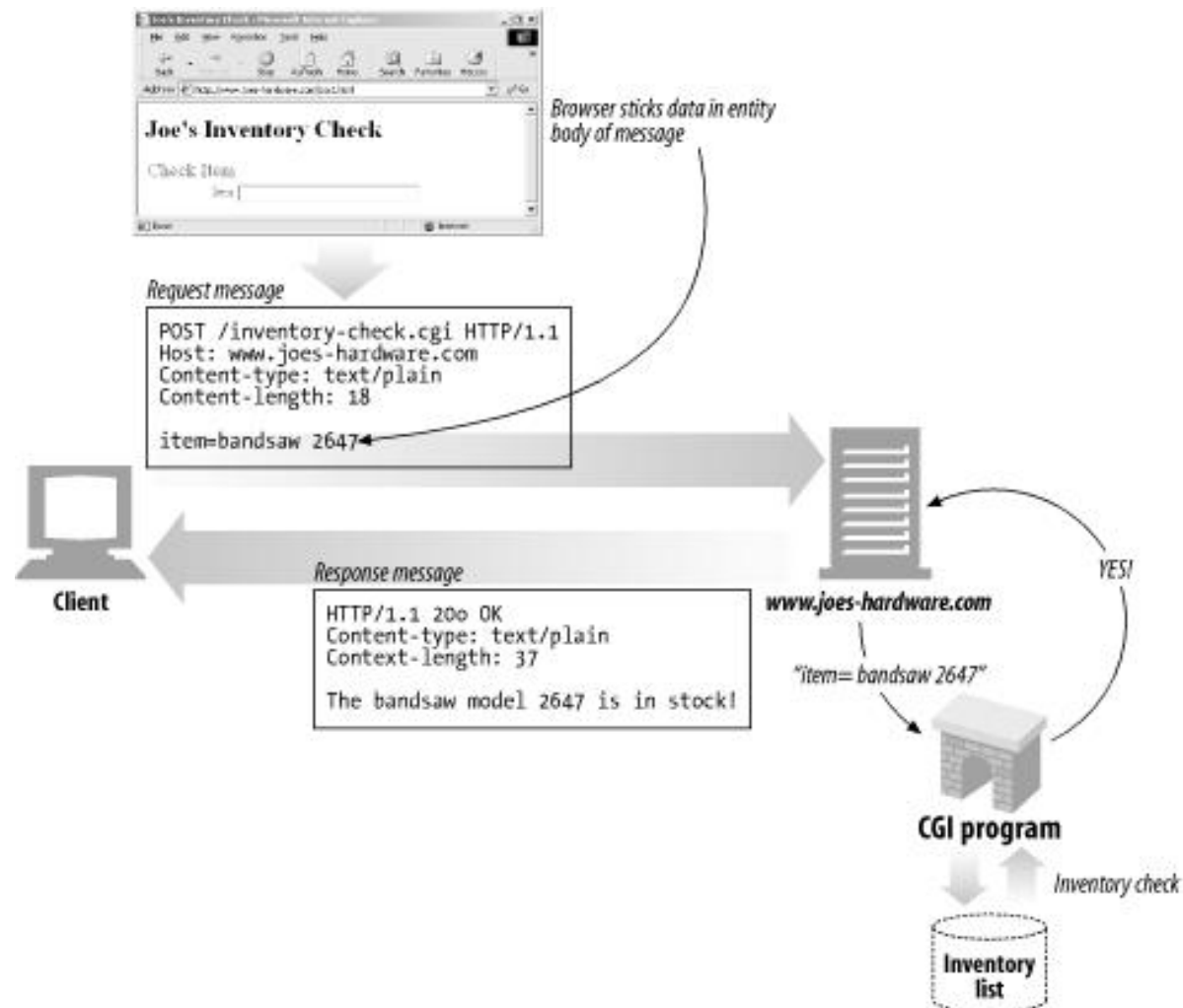
# PUT

- The PUT method writes documents to a server, in the inverse of the way that GET reads documents from a server.



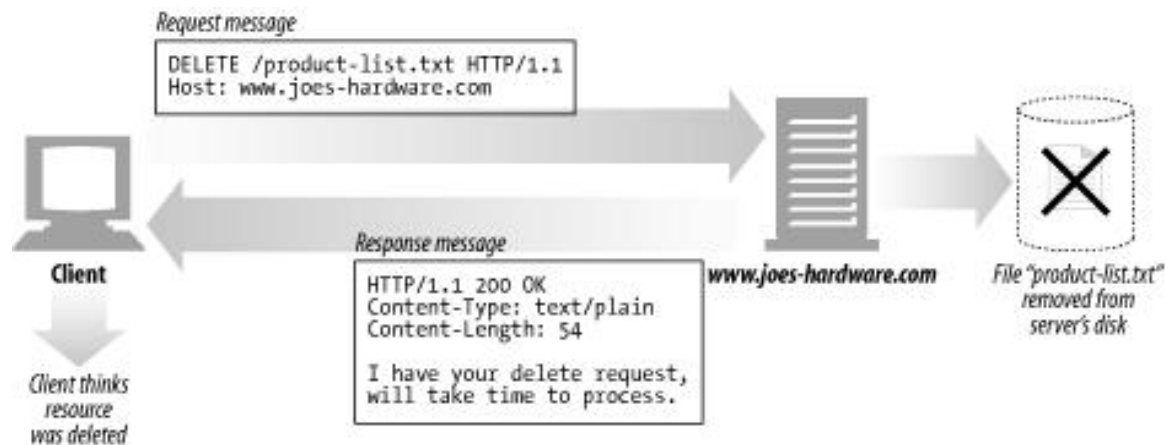
# POST

- The POST method was designed to send input data to the server.
- In practice, it is often used to support HTML forms.



# DELETE

- It asks the server to delete the resources specified by the request URL.



# Status Codes

- Every HTTP response message comes back with a status code.
- The **status code** is a three-digit numeric code that tells the client if the request succeeded, or if other actions are required.

Some common HTTP status codes	
HTTP status code	Description
200	OK. Document returned correctly.
302	Redirect. Go someplace else to get the resource.
404	Not Found. Can't find this resource.



# HTTP Headers

- General headers
  - They serve general purposes that are useful for clients, servers, and other applications to supply to one another.
  - Eg. Date: Tue, 3 Oct 1974 02:16:00 GMT
- Request headers
  - They provide extra information to servers, such as what type of data the client is willing to receive.
  - Eg **Accept**: image/gif, image/jpeg, text/html will tell the server that the client will accept any kind of data
- Response headers
  - provide information to the client
  - Eg. Server: Tiki-Hut/1.0 – the type of server
- Entity headers
  - refer to headers that deal with the entity body.
  - Eg. Content-Type: text/html; charset=iso-latin-1
- Extension headers
  - are nonstandard headers that have been created by application developers but not yet added to the approved HTTP specification.

# Entity Bodies

- The third part of an HTTP message is the optional entity body.
- HTTP messages can carry many kinds of digital data:
  - images,
  - video,
  - HTML documents,
  - software applications,
  - credit card transactions,
  - electronic mail, and so on.

# Client Identification

- Http began as a stateless, request/response protocol
- Later client Identification was needed to
  - *Personal greetings*
  - *Targeted recommendations*
  - *Administrative information on file*
  - *Session tracking*

## Cont'd . . . **Client Identification**

- Some of the techniques used to identify users in HTTP :
- **HTTP headers** that carry information about user identity
- **Client IP address** tracking, to identify users by their IP addresses
- **User login**, using authentication to identify users
- **Fat URLs**, a technique for embedding identity in URLs
- **Cookies**, a powerful but efficient technique for maintaining persistent identity

# Using HTTP Headers

HTTP headers carry clues about users		
Header name	Header type	Description
From	Request	User's email address
User-Agent	Request	User's browser software
Referer	Request	Page user came from by following link
Authorization	Request	Username and password (discussed later)
Client-ip	Extension (Request)	Client's IP address (discussed later)
X-Forwarded-For	Extension (Request)	Client's IP address (discussed later)
Cookie	Extension (Request)	Server-generated ID label (discussed later)

- Since each user would have a different **email address**
  - but due to dishonest servers collecting email addresses and using them for junk mail distribution it is not mostly used.
- The **User-Agent header** is useful for customizing content to particular browsers, but not to identify a particular user.
- **Referer** – useful to better understand user browsing behavior and user interests.
- **The From, User-Agent, and Referer headers are insufficient for dependable identification purposes.**

# Using Client IP Address

- Using the client IP address to identify the user has limitations:
  - Client IP addresses describe only the computer being used, not the user.
  - Many Internet service providers dynamically assign IP addresses to users when they log in.
  - HTTP proxies and gateways typically open new TCP connections to the origin server. The web server will see the IP address of the proxy server instead of that of the client.

# Using User Login

- By asking the user to authenticate (log in) with a username and password.
- HTTP includes a built-in mechanism to pass username information to web sites, using the
  - WWW-Authenticate and
  - Authorization headers.
- Once logged in, the browsers continually send this login information with each request to the site, so the information is always available.

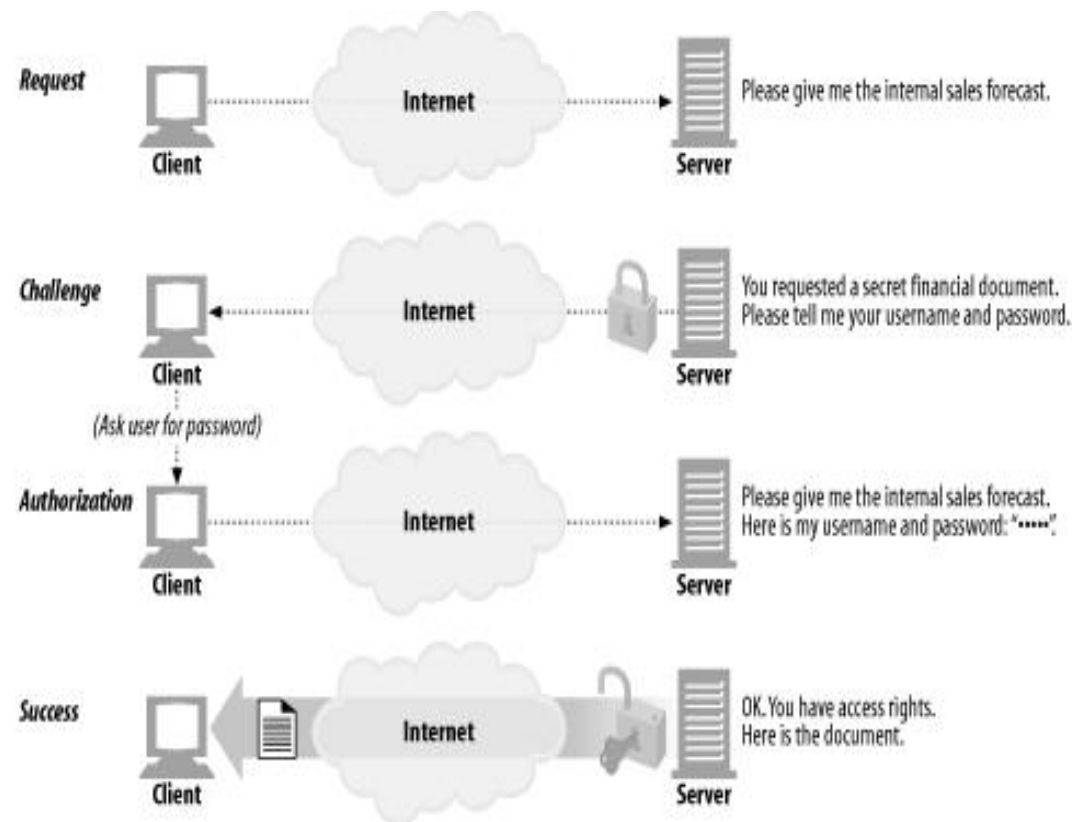
# Using Fat URLs

- By keeping track of user identity by generating special versions of each URL for each user.
- a real URL is extended by adding some state information to the start or end of the URL path.
- As the user browses the site, the web server dynamically generates hyperlinks that continue to maintain the state information in the URLs.
- Fat URLs have some limitations:
  - *Ugly URLs*
  - *Can't share URLs*
  - *Breaks caching*
  - *Extra server load*
  - *Not persistent across sessions*

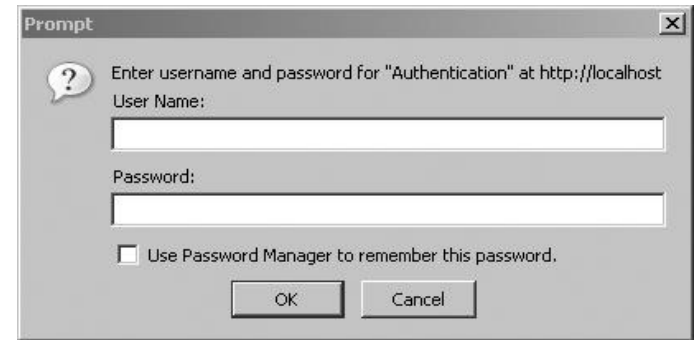


# Authentication

- Authentication means showing some proof of your identity.
- HTTP provides a native challenge/response framework to make it easy to authenticate users.



# Authentication



- The HTTP protocol offers a fairly effective means for user authentication. A typical authentication scenario proceeds like this:
  - i. The client requests a restricted resource.
  - ii. The server responds to this request with a 401 (Unauthorized access) response message.
  - iii. The client (browser) recognizes the 401 response and produces a pop-up authentication prompt similar to the one shown above.
  - iv. The user-supplied credentials (namely, the username and password) are sent back to the server for validation. If the user supplies correct credentials, access is granted; otherwise it's denied.
  - v. If the user is validated, the browser stores the authentication information within its authentication cache. This cache information remains within the browser until the cache is cleared, or until another 401 server response is sent to the browser.
- Both the supplied username and password are included in this traffic, both **unencrypted**.

# Web browser

- To allow all browsers to understand all Web pages they are written in a standardized language called HTML
- Not all pages contain HTML.
- Rather than making the browsers larger and larger by building in interpreters for a rapidly growing collection of file types, most browsers have chosen a more general solution.
- There are two possibilities: [plug-ins](#) and [helper applications](#).
  - A plug-in is a code module that the browser fetches from a special directory on the disk and installs as an extension to itself.
    - eg. download managers,
  - A helper application- is a complete program, running as a separate process.
    - MS word, PDF reader

# Questions