

# COMP 484 - Web Engineering I

Instructor:

Tania Babakhanlou

Reference:

Computer Networks, Tanenbaum

# Application Layer

## Chapter 7

- DNS – Domain Name System
- The Web and html

Revised: August 2011

# The Application Layer

- The application layer contains **programs** that make use of the network. It uses **transport services** to build distributed applications
- Application Layer Protocol standardizes **communication**
- Contains a variety of protocols. Example: HTTP (HyperText Transfer Protocol)
- HTTP: the basis for the World Wide Web. When a browser wants a Web page, it sends the name of the page it wants to the server hosting the page using HTTP. The server then sends the page back.
- The HTTP protocol defines how to use the network to serve the web page to the client.
- The user interface for HTTP Protocol is the Web browser.

Application
Transport
Network
Link
Physical

# DNS – Domain Name System

Example of an application layer protocol.

The DNS resolves high-level human readable names for computers to low-level IP addresses

Example: Host Name : csun.edu ⇔ IP Address : 130.166.238.195

(use "ping" or "nslookup" command)

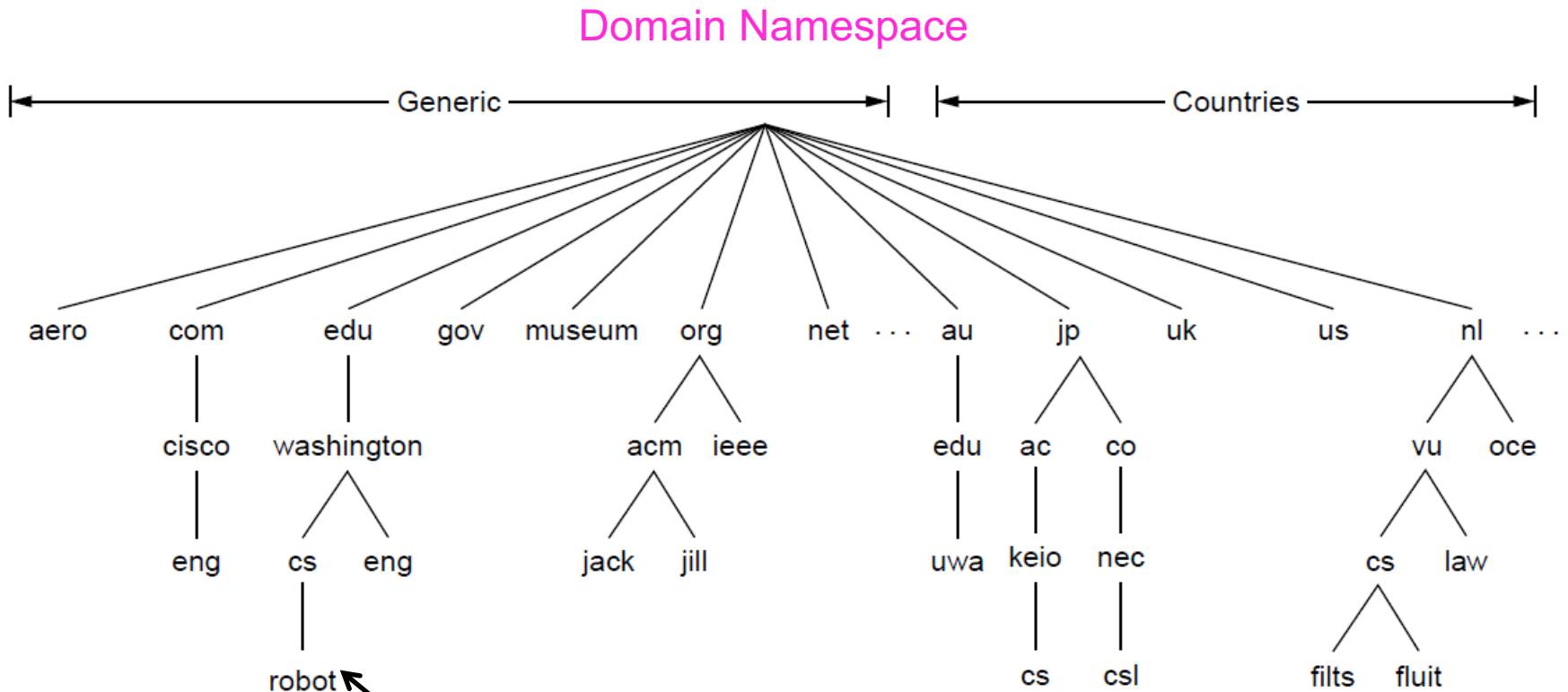
We will discuss:

- DNS name space
- Domain Resource records
- Name servers

# The DNS Name Space (1)

DNS namespace is hierarchical from the (unnamed) root down

- Different parts delegated to different organizations
- Over 250 top-level domains
- Subdomains, then partitioned more
- A tree structure



The computer *robot.cs.washington.edu*

# The DNS Name Space (2)

- **Generic top-level domains:**  
are controlled by ICANN who appoints registrars to run them
- **Country top-level domains:**  
one entry for each country.  
US, UK, AU, ...

Domain	Intended use	Start date	Restricted?
com	Commercial	1985	No
edu	Educational institutions	1985	Yes
gov	Government	1985	Yes
int	International organizations	1988	Yes
mil	Military	1985	Yes
net	Network providers	1985	No
org	Non-profit organizations	1985	No
aero	Air transport	2001	Yes
biz	Businesses	2001	No
coop	Cooperatives	2001	Yes
info	Informational	2002	No
museum	Museums	2002	Yes
name	People	2002	No
pro	Professionals	2002	Yes
cat	Catalan	2005	Yes
jobs	Employment	2005	Yes
mobi	Mobile devices	2005	Yes
tel	Contact details	2005	Yes
travel	Travel industry	2005	Yes
xxx	Sex industry	2010	No

# The DNS Name Space (2)

- Each domain is named by the path upward from it to the (unnamed) root.
- components are separated by periods
- A **Named Domain** refers to a specific node in the tree and all the nodes under it.
- Domain names are case-insensitive
- There is no rule against registering under multiple top-level domains:  
wikipedia.org, wikipedia.com
- Each domain controls how it allocates the domains under it.
- To create a new domain, permission is required of the domain in which it will be included.
- Naming follows organizational boundaries, not physical networks

# Domain Resource Records

Every domain, whether it is a single host or a top-level domain, can have a **set of resource records** associated with it. These records are the **DNS database**. They are accessed during DNS lookups.

The key resource records in the namespace are **IP addresses** (A/AAAA) (most common for a single host) and **name servers (NS)**, but there are others too (e.g., MX)

Type	Meaning	Value
SOA	Start of authority	Parameters for this zone
A	IPv4 address of a host	32-Bit integer
AAAA	IPv6 address of a host	128-Bit integer
MX	Mail exchange	Priority, domain willing to accept email
NS	Name server	Name of a server for this domain
CNAME	Canonical name	Domain name
PTR	Pointer	Alias for an IP address
SPF	Sender policy framework	Text encoding of mail sending policy
SRV	Service	Host that provides it
TXT	Text	Descriptive ASCII text



# Domain Resource Records

; Authoritative data for cs.vu.nl

cs.vu.nl.	86400	IN	SOA	star boss (9527,7200,7200,241920,86400)	
cs.vu.nl.	86400	IN	MX	1 zephyr	← Mail servers for @cs.vu.nl
cs.vu.nl.	86400	IN	MX	2 top	
cs.vu.nl.	86400	IN	NS	star	← Name server
star	86400	IN	A	130.37.56.205	
zephyr	86400	IN	A	130.37.20.10	← IP addresses of computers
top	86400	IN	A	130.37.20.11	
www	86400	IN	CNAME	star.cs.vu.nl	← Alias for www.cs.vu.nl
ftp	86400	IN	CNAME	zephyr.cs.vu.nl	
flits	86400	IN	A	130.37.16.112	← IP addresses flits.cs.vu.nl
flits	86400	IN	A	192.31.231.165	
flits	86400	IN	MX	1 flits	← Mail servers for @flits.cs.vu.nl
flits	86400	IN	MX	2 zephyr	
flits	86400	IN	MX	3 top	
rowboat		IN	A	130.37.56.201	← IP entry for a computer: rowboat
		IN	MX	1 rowboat	
		IN	MX	2 zephyr	← Mail gateways for rowboat
little-sister		IN	A	130.37.62.23	
laserjet		IN	A	192.31.231.216	

A portion of a possible **DNS database** for cs.vu.nl domain

The **Format** of a record: DomainName Timetolive Class Type Value

# Domain Resource Records

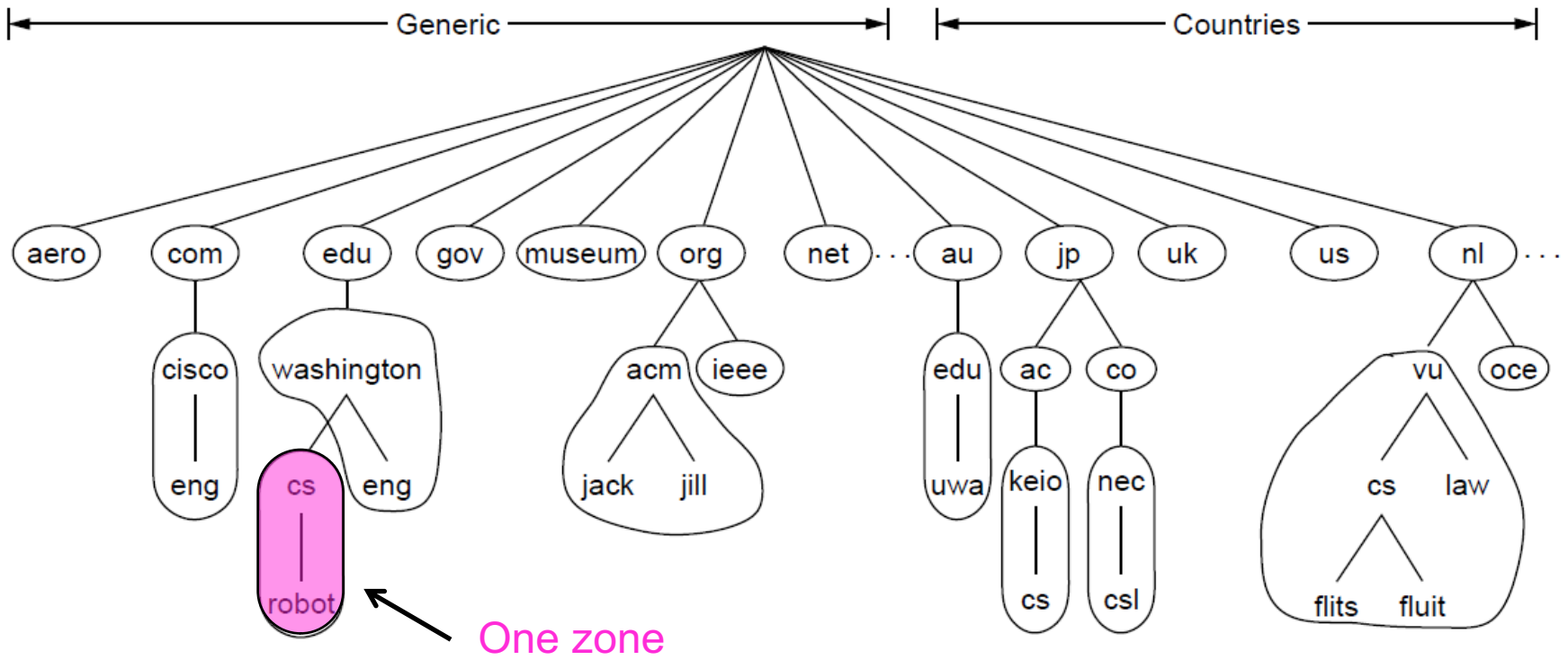
- **A (Address) record:** a 32-bit IPv4 address of an interface for some host. (maps a name to an IP)
- **AAAA record :** 128-bit IPv6 address.
- Every Internet host must have at least one IP address. DNS can return multiple addresses for a single name if multiple A records exist.
- **MX record:** The name of the host accepting **email** for the specified domain. Each MX record points to an email server that's configured to process mail for that domain
- **NS record:** A name server for the domain or subdomain. A host that has a copy of the database for a domain and used as part of the process to **look up names**
- **CNAME record:** allows aliases to be created. A mechanism to replace one string by another (maps a name to another name)

# Domain Resource Records

- **SOA record(Start of Authority):** provides the name of the primary source of information about the name server's zone, the email address of its administrator, a unique serial number, and various flags and timeouts. A delegation cut off point of a domain name zone from its parent. A DNS zone is the part of a domain for which an individual DNS server is responsible. Each zone contains a single SOA record.
- **PTR record (Pointer records):** Like CNAME, PTR points to another name. Nearly always used for reverse lookups (IP -> name). Associates a name with an IP address. (An A record should exist for every PTR record)
- **SRV record (Service record):** allows a host to be identified for a given service in a domain. For example, the Web server for cs.washington.edu could be identified as cockatoo.cs.washington.edu.
- **TXT record(text record):** used to associate arbitrary info with the server. Human readable information or to encode machine readable information, typically the SPF information
- **SPF(Sender Policy Framework) record:** lets a domain encode information about what machines in the domain for example will send mail to the rest of the Internet. A type of TXT record

# Name Servers (1)

Name servers contain data for portions of the name space called zones (circled).



# Name Servers (2)

Finding the IP address for a given hostname is called resolution and is done with the DNS protocol.

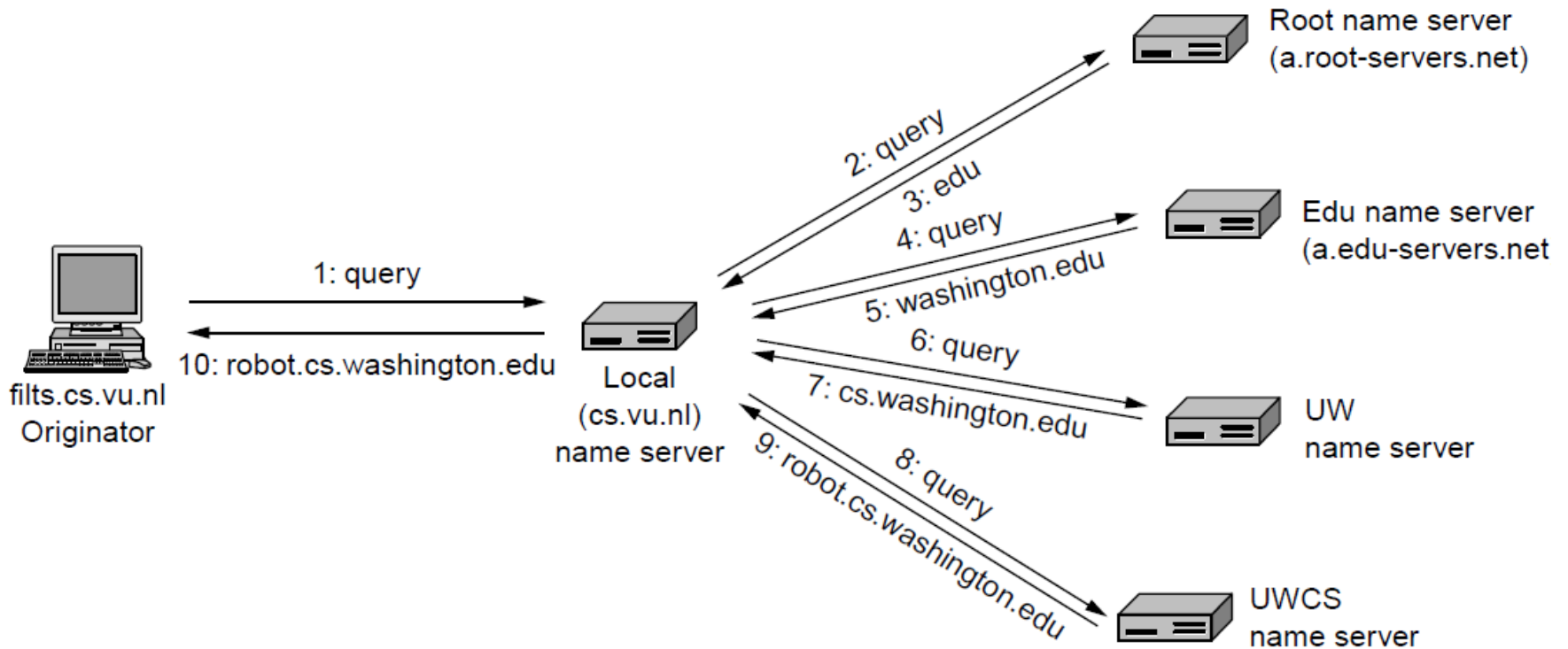
Name Resolution:

- Computer requests local name server to resolve
- Local name server asks the root name server
- Root returns the name server for a lower zone
- Continue down **zones** until name server can answer

DNS protocol:

- Runs on UDP port 53, retransmits lost messages
- Caches name server answers for better performance

# Name Servers (3)



Example of a computer looking up the IP for a name

# The World Wide Web

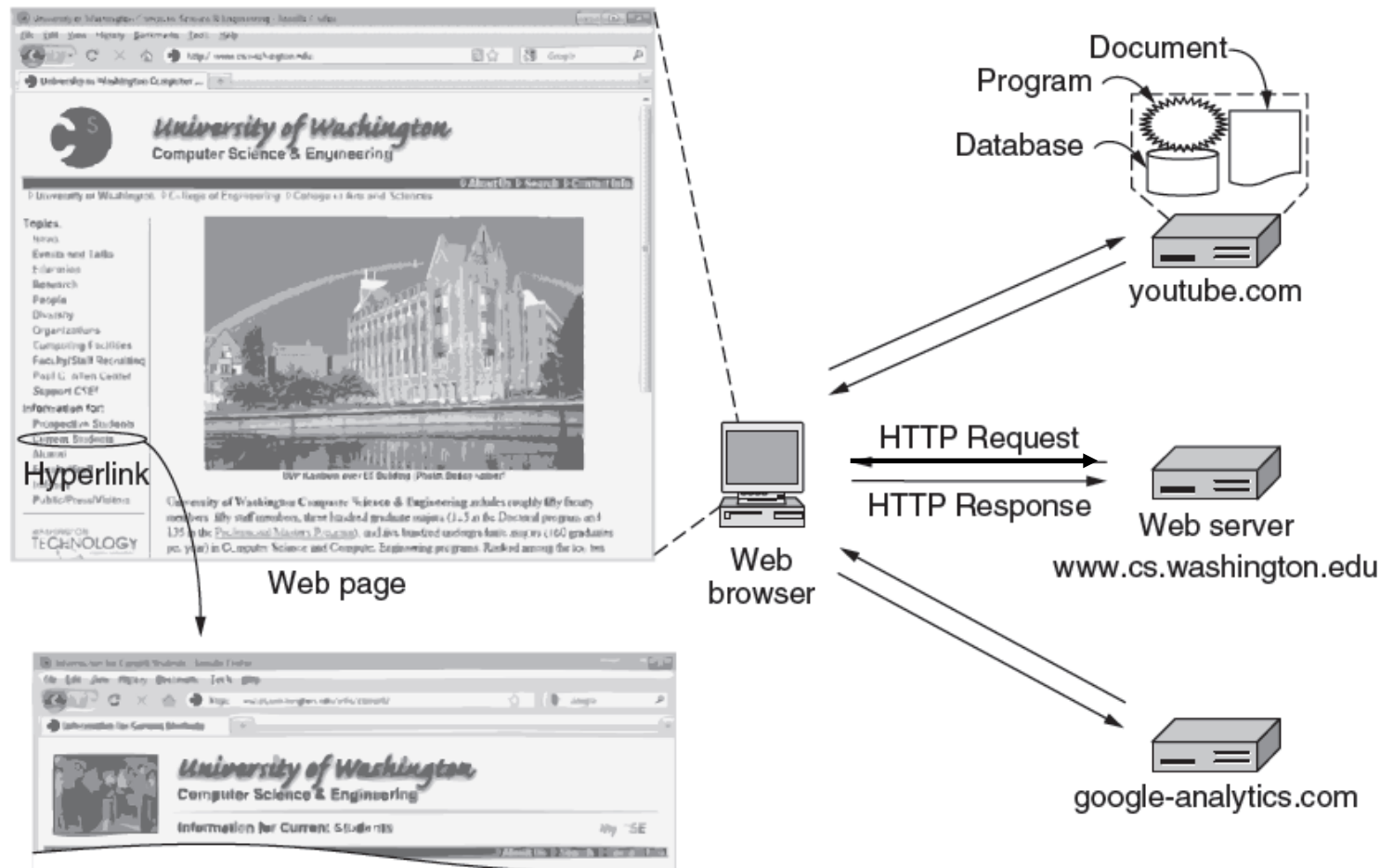
- Architectural overview »
- Static Web pages »
- Dynamic pages and Web applications »
- HTTP – HyperText Transfer Protocol »

# Architectural Overview

HTTP (Hypertext transfer protocol) transfers pages from servers to browsers.

Local vs. remote servers. (localhost, 127.0.0.1, URL)

Example: `http://127.0.0.1/lab/examples/ch02/form.html`





# Architectural Overview

Pages are named with URLs (Uniform Resource Locators)

- Example: <http://www.phdcomics.com/comics.php>

  
Protocol                      Server                      Page on server

Our  
focus →

Name	Used for	Example
http	Hypertext (HTML)	<a href="http://www.ee.uwa.edu/~rob/">http://www.ee.uwa.edu/~rob/</a>
https	Hypertext with security	<a href="https://www.bank.com/accounts/">https://www.bank.com/accounts/</a>
ftp	FTP	<a href="ftp://ftp.cs.vu.nl/pub/minix/README">ftp://ftp.cs.vu.nl/pub/minix/README</a>
file	Local file	<a href="file:///usr/suzanne/prog.c">file:///usr/suzanne/prog.c</a>
mailto	Sending email	<a href="mailto:JohnUser@acm.org">mailto:JohnUser@acm.org</a>
rtsp	Streaming media	<a href="rtsp://youtube.com/montypython.mpg">rtsp://youtube.com/montypython.mpg</a>
sip	Multimedia calls	<a href="sip:eve@adversary.com">sip:eve@adversary.com</a>
about	Browser information	<a href="about:plugins">about:plugins</a>

Common URL protocols

# Architectural Overview

Steps a client (browser) takes to follow a hyperlink:

- Determine the protocol (HTTP)
- Ask DNS for the IP address of server
- Make a TCP connection to server
- Send request for the page; server sends it back
- Fetch other URLs as needed to display the page
- Close idle TCP connections

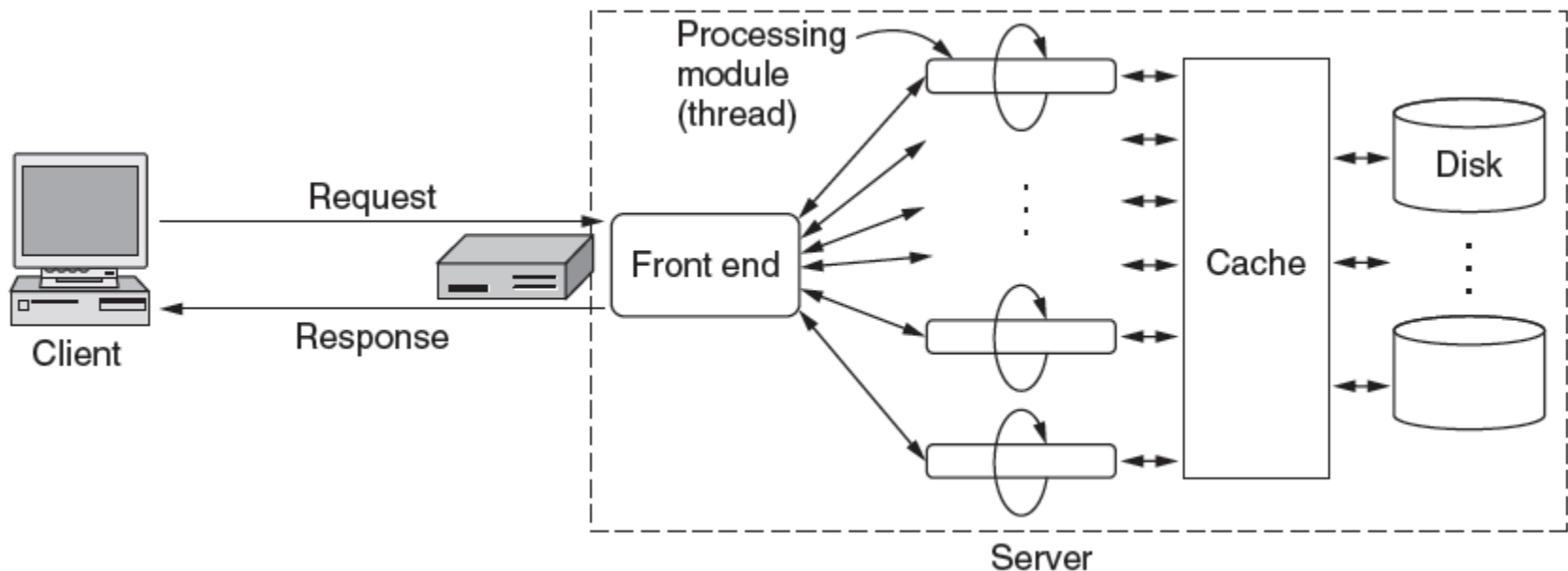
Steps a server takes to serve pages:

- Accept a TCP connection from client
- Get page request and map it to a resource (e.g., file name)
- Get the resource (e.g., file from disk)
- Send contents of the resource to the client.
- Release idle TCP connections

# Architectural Overview

To scale performance, Web servers can use:

- Caching, multiple threads, and a front end



# Architectural Overview

Server steps, revisited:

- Accept a TCP connection from client
- Resolve name of Web page requested
- Perform **access control** on the Web page
- Check the **cache**
- Fetch requested page from disk or run program
- Determine the rest of the response
- Return the response to the client
- Make an entry in the server **log**
- Release idle TCP connections

# Architectural Overview

Cookies support stateful client/server interactions

- Server sends cookies (state) with page response
- Client stores cookies across page fetches
- Client sends cookies back to server with requests

Domain	Path	Content	Expires	Secure
toms-casino.com	/	CustomerID=297793521	15-10-10 17:00	Yes
jills-store.com	/	Cart=1-00501;1-07031;2-13721	11-1-11 14:22	No
aportal.com	/	Prefs=Stk:CSCO+ORCL;Spt:Jets	31-12-20 23:59	No
sneaky.com	/	UserID=4627239101	31-12-19 23:59	No

Examples of cookies

# Static Web Pages (1)

Static Web pages are simply files

- Have the same contents for each viewing

Can be visually rich and interactive nonetheless:

- HTML that mixes text and images
- Forms that gather user input
- Style sheets that tailor presentation
- Vector graphics, videos, and more . . .

# Static Web Pages (2)

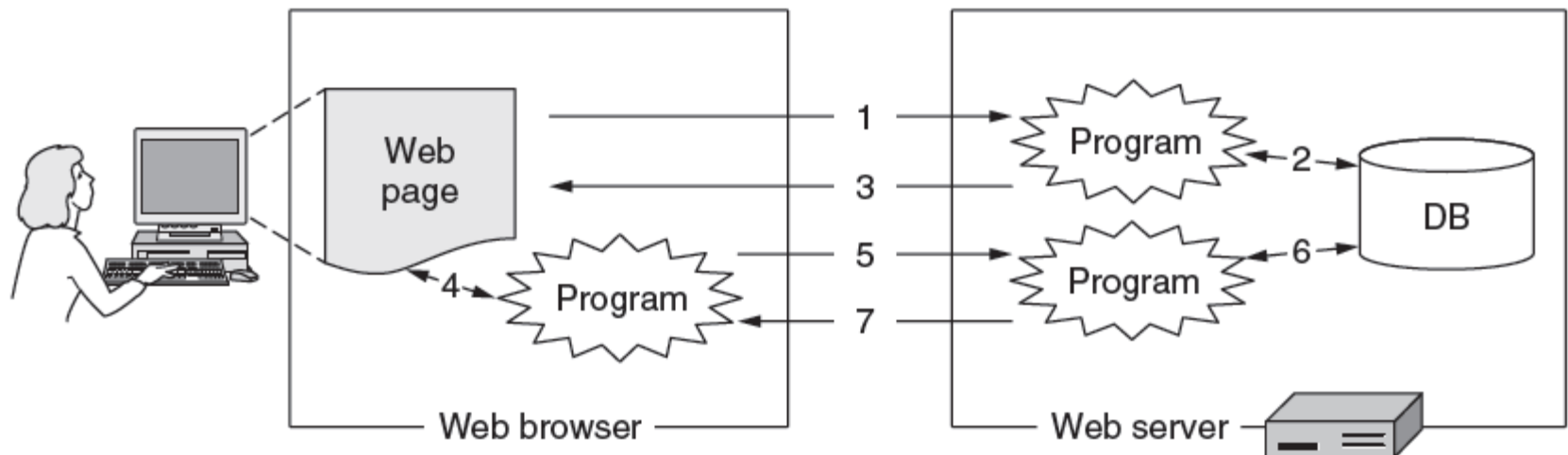
## Progression of features through HTML 5.0

Item	HTML 1.0	HTML 2.0	HTML 3.0	HTML 4.0	HTML 5.0
Hyperlinks	X	X	X	X	X
Images	X	X	X	X	X
Lists	X	X	X	X	X
Active maps & images		X	X	X	X
Forms		X	X	X	X
Equations			X	X	X
Toolbars			X	X	X
Tables			X	X	X
Accessibility features				X	X
Object embedding				X	X
Style sheets				X	X
Scripting				X	X
Video and audio					X
Inline vector graphics					X
XML representation					X
Background threads					X
Browser storage					X
Drawing canvas					X

# Dynamic Pages & Web Applications

Dynamic pages are generated by programs running at the **server** (with a database) and the **client**

- E.g., PHP at server, JavaScript at client
- Pages vary each time like using an application





# Dynamic Pages & Web Applications

JavaScript program  
(client side)  
produces result  
page in the browser

```
<html>
<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>
```

First page with form,  
gets input and calls  
program above

```
<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 Pages & Web Applications


Web page that gets form input and calls a server program

```
<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>
```

PHP server program that creates a custom Web page (exists on server side only)

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

PHP calls

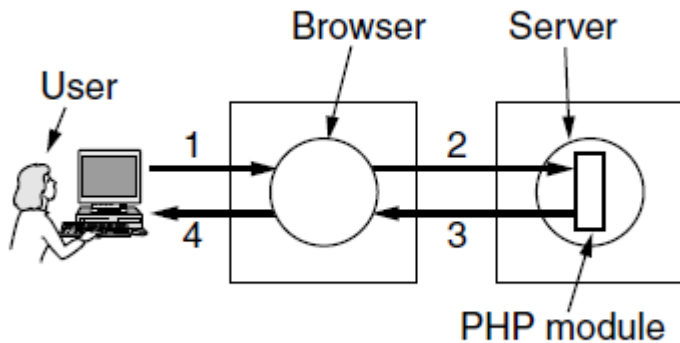


Resulting Web page for inputs “Barbara” and “32” (this is sent to the client)

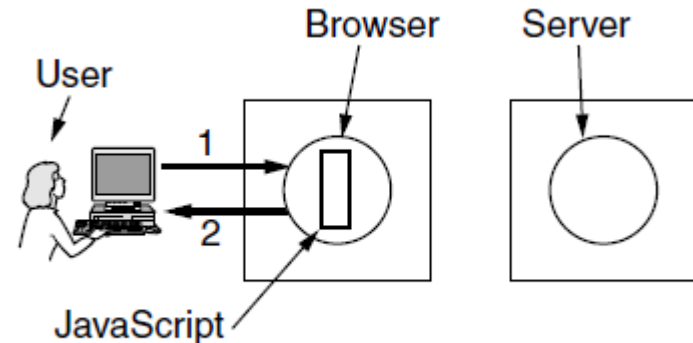
```
<html>
<body>
<h1> Reply: </h1>
Hello Barbara.
Prediction: next year you will be 33
</body>
</html>
```

# Dynamic Pages & Web Applications

The difference between server and client programs



Server-side scripting with PHP



Client-side scripting with JavaScript

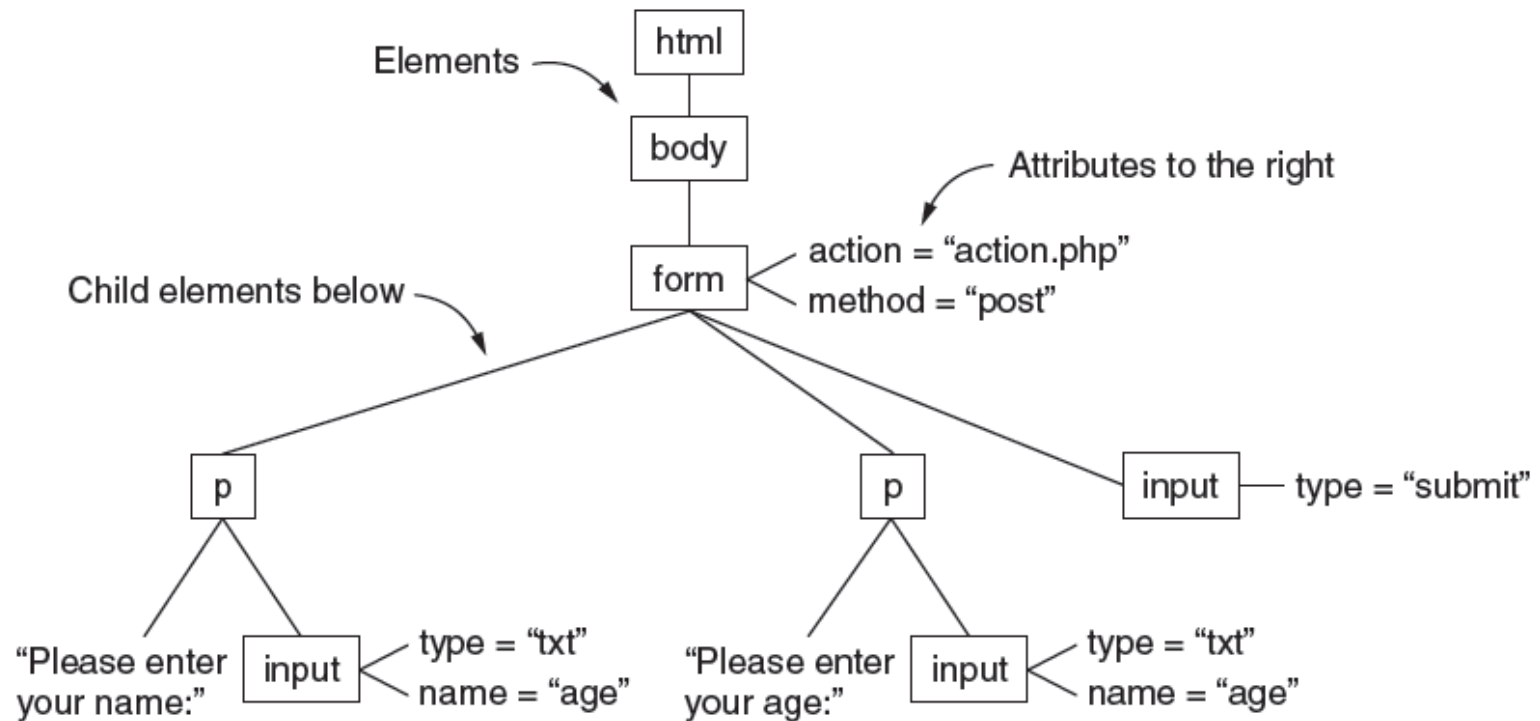
# Dynamic Pages & Web Applications

Dynamic web applications use a set of technologies that work together, e.g.

- AJAX: Asynchronous JavaScript and Xml
- HTML: present information as pages.
- DOM: change parts of pages while they are viewed.
- Scripts: JavaScript, PHP
- XML: let programs exchange data with the server. Ajax uses a asynchronous way to send and retrieve XML data.

# Dynamic Pages & Web Applications

The DOM (Document Object Model) tree is an application programming interface that represents Web pages(HTML documents) as a structure that programs can alter



# Dynamic Pages & Web Applications

- The root is the html element. It is the parent of the body element.
- Form has Attributes : Action, Method
- Leaves: elements or literals(text strings)
- DOM model provides programs with a straightforward way to change parts of the page. Only the node that contains the change needs to be replaced.

# Dynamic Pages & Web Applications

XML: a language for specifying structured content.

XML captures document structure, not presentation/formatting like HTML does. Ex:

- No defined tags for XML (vs. html)
- User can define own tags
- XSD: XML Schema Definition / validation

```
<?xml version="1.0" ?>
<book_list>
  <book>
    <title> Human Behavior and the Principle of Least Effort </title>
    <author> George Zipf </author>
    <year> 1949 </year>
  </book>
  <book>
    <title> The Mathematical Theory of Communication </title>
    <author> Claude E. Shannon </author>
    <author> Warren Weaver </author>
    <year> 1949 </year>
  </book>
  <book>
    <title> Nineteen Eighty-Four </title>
    <author> George Orwell </author>
    <year> 1949 </year>
  </book>
</book_list>
```

# HTTP (1)

HTTP (HyperText Transfer Protocol) is a request-response protocol in application layer, that runs on top of TCP in transport layer

- Fetches pages from server to client
- Server process usually runs on port 80 (listens to port 80)
- Headers are given in readable ASCII
- Protocol has support for pipelining requests (multiple requests on a single TCP connection without waiting for responses)
- Protocol has support for caching



# HTTP (3)

HTTP has several request methods (RFC 2616).

	Method	Description
Fetch a page →	GET	Read a Web page
	HEAD	Read a Web page's header
Used to send input data to a server program →	POST	Append to a Web page
	PUT	Store a Web page
	DELETE	Remove the Web page
	TRACE	Echo the incoming request
	CONNECT	Connect through a proxy
	OPTIONS	Query options for a page

# HTTP (4)

Response codes tell the client how the request went:

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

# HTTP (5)

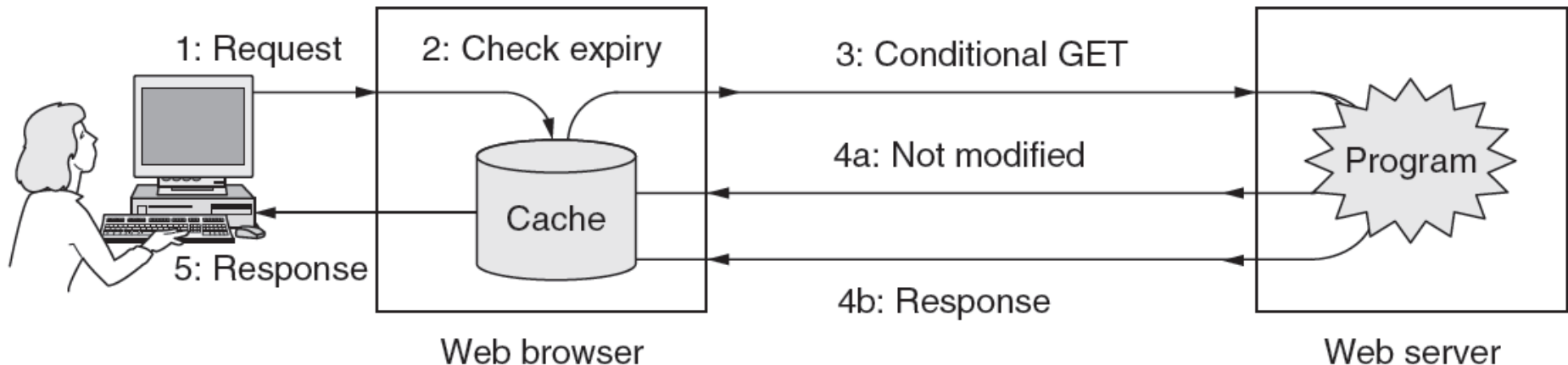
Many headers carry key information:

Function	Example Headers
Browser capabilities (client → server)	User-Agent, Accept, Accept-Charset, Accept-Encoding, Accept-Language
Caching related (mixed directions)	If-Modified-Since, If-None-Match, Date, Last-Modified, Expires, Cache-Control, ETag
Browser context (client → server)	Cookie, Referrer, Authorization, Host
Content delivery (server → client)	Content-Encoding, Content-Length, Content-Type, Content-Language, Content-Range, Set-Cookie

# HTTP (6)

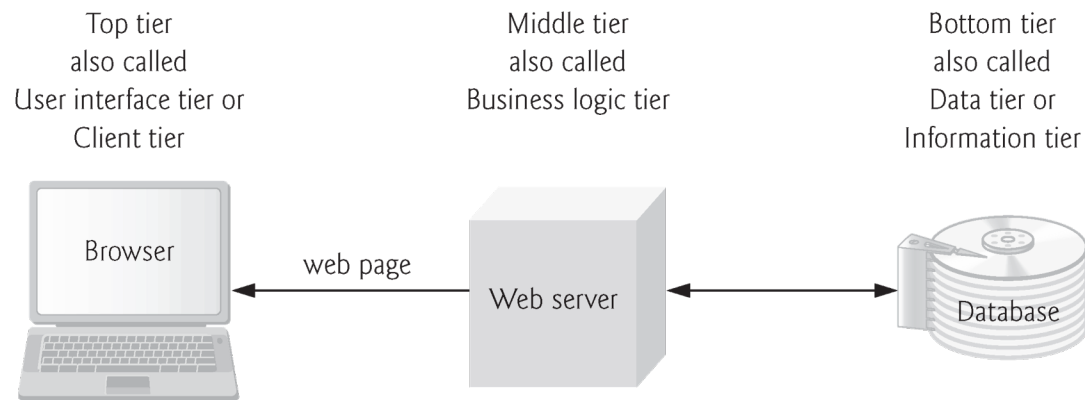
HTTP caching checks to see if the browser has a known fresh copy, and if not if the server has updated the page

- Uses a collection of headers for the checks
- Can include further levels of caching (e.g., proxy)
- **Conditional** Get request includes validator headers such as "If-Modified-Since" and the response can be HTTP 304 meaning the resource has not been modified since the previous GET, so the resource is not returned.



# 1.7 Multitier Application Architecture

Web-based applications are often **multitier applications** (***n-tier applications***) that divide functionality into separate **tiers** (logical groupings of functionality)



**Fig. 1.10** | Three-tier architecture.

the basic structure of a **three-tier web-based application**

# 1.7 Multitier Application Architecture (cont.)

The **bottom tier** (the data tier or the information tier)

- maintains the application's data.
- stores data in a relational database management system (RDBMS).

The **middle tier**: including the web server, implements business logic,

- controller logic and presentation logic to control interactions between the application's clients and its data.
- acts as an intermediary between data in the information tier and the application's clients.
- The **controller logic** processes client requests (such as requests to view a product catalog) and retrieves data from the database.
- The **presentation logic** processes data from the information tier and presents the content(HTML doc) to the client
- **Business logic** enforces **business rules** and ensures that data is reliable before the application updates a database or presents data to users. Business rules dictate how clients access data and how applications process data.

# 1.7 Multitier Application Architecture (cont.)

The **top tier**, or client tier, is the application's user interface, which gathers input and displays output.

Users interact directly with the application through the user interface, which is typically a web browser or a mobile device.

In response to user actions (e.g., clicking a hyperlink), the client tier interacts with the middle tier to make requests and to retrieve data from the information tier.

The client tier then displays the data retrieved for the user.



# Experimenting with HTTP

- Telnet: An application layer protocol used for remote host connection (on a given port) over a TCP/IP network and to run commands on the remote host.
- Telnet client software is needed on the client machine and telnet server is needed on the server or the remote host.
- Telnet is not secure. Telnet data (including passwords) is sent in clear text. TSL is an extension for telnet making it secure.
- We can use telnet to connect to a server and download HTML files using HTTP protocol.

# Experimenting with HTTP

Use terminal to communicate with a web server, through a TCP connection to port 80 on the server

```
telnet www.ietf.org 80  
GET /rfc.html HTTP/1.1  
Host: www.ietf.org
```

*=====*

```
telnet www.ietf.org 80  
GET /HTTP/1.1  
Host: www.ietf.org
```

1. Starts up a telnet (TCP) connection to port 80 of a specific server
2. GET command: request to obtain a resource from the server
  1. Path of the URL
  2. The protocol (HTTP) and version (1.1)
3. Mandatory Host headers in 1.1 (host: important since multiple domains can be hosted on a single IP)
4. (The User-Agent and other options can be specified on subsequent lines. Example: "User-Agent: Mozilla/4.0")
5. A blank line following the last header is mandatory, tells the server that there are no more request headers

# Experimenting with HTTP

Response:

HTTP status code: 200 success

HTTP headers: additional info about the data

Content type, cookie, programs to use, how the browser should interpret the data,...

A blank line indicating the end of the headers

Then HTML document

```
HTTP/1.1 200 OK
Date: Mon, 19 Sep 2016 07:54:42 GMT
Content-Type: text/html
Transfer-Encoding: chunked
Connection: keep-alive
Set-Cookie:
__cfduid=d5a27b2bfb7af9dc05ac2ace0421c62951474271682;
expires=Tue, 19-Sep-17 07:54:42 GMT; path=/; domain=.ietf.org;
HttpOnly
Last-Modified: Fri, 08 Jan 2016 17:42:02 GMT
ETag: W/"3743-528d61ae6cb21-gzip"
Vary: Accept-Encoding
Strict-Transport-Security: max-age=31536000
X-Frame-Options: SAMEORIGIN
X-Xss-Protection: 1; mode=block
X-Content-Type-Options: nosniff
CF-Cache-Status: HIT
Expires: Mon, 19 Sep 2016 11:54:42 GMT
Cache-Control: public, max-age=14400
Server: cloudflare-nginx
CF-RAY: 2e4b7882e5d15378-LAX

3743
<!DOCTYPE HTML PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
```

# Experimenting with HTTPS

HTTPS:

- To request a page from a secure (SSL) server on port 443 you can use **openssl** instead of telnet. Example:
- `openssl s_client -connect google.com:443 -tls1_2`
- `openssl s_client -connect google.com:443 -ssl3`
- Ciphers: SSLv3, TLSv1, TLSv1.1, TLSv1.2

HTTPS: HTTP Secure (over SSL **or** TSL)

*Request:*

`openssl s_client -connect www.facebook.com:443`

`GET / HTTP/1.1`

`Host: www.facebook.com`

=====

`openssl s_client -connect www.facebook.com:443`

`GET / HTTP/1.1`

`Host: www.facebook.com`

`User-Agent: Mozilla/4.0`

Testing if the HTTPS certificate has been installed properly on the server.

# Experimenting with HTTPS

## Response:

Certificate info and issuer, Server Certificate, SSL-session, SSL handshake, SSL-session Protocol and version (TLS, SSL),

And HTTP response.

```
[tbabakhanlou:~]$ openssl s_client -connect www.facebook.com:443
CONNECTED(00000003)
depth=1 /C=US/O=DigiCert Inc/OU=www.digicert.com/CN=DigiCert High Assurance CA-3
verify error:num=20:unable to get local issuer certificate
verify return:0
---
Certificate chain
 0 s:/C=US/ST=CA/L=Menlo Park/O=Facebook, Inc./CN=*.facebook.com
  i:/C=US/O=DigiCert Inc/OU=www.digicert.com/CN=DigiCert High Assurance CA-3
 1 s:/C=US/O=DigiCert Inc/OU=www.digicert.com/CN=DigiCert High Assurance CA-3
  i:/C=US/O=DigiCert Inc/OU=www.digicert.com/CN=DigiCert High Assurance EV Root CA
---
Server certificate
-----BEGIN CERTIFICATE-----
MIIHtzCCBjegAwIBAgIQDnQ+EartPbA34VVvKGw/4TANBgkqhkiG9w0BAQUFADBm
MQswCQYDVQQGEwJVUzEVMBMGA1UEChMMRGlnaUNlcnQgSW5jMRkwFwYDVQQLExB3
d3cuZGlnaWNlcnQuY29tMSUwIiwYDVQDEwEaWdpQ2VydCBIdWdoIEFzc3VyYW5j
ZSBDOQ0zMB4XDTE1MDQxMDAwMDAwMFoXDTE2MTIzMDEyMDAwMFowYTELMAGGA1UE
BhMCVVMxQzAJBgNVBAGTAkNBMRMwEQYDVQQHEwZpYXJrMRcwFQYDVQQK
Ew5GYWNlYm9vaywgSW5jLjEXMBUGA1UEAwOKi5mYWNlYm9vay5jb20wggEiMA0G
CSqGSIb3DQEBAQUAA4IBDwAwggEKAoIBAQC/wTWu0yvMMtGC09/WYnefzxhC8jnu
EB6h3eJGA7FmZ+G1DoneEo/OMcsCid+n+t7zd/A6+ZVudcFcaIyqJ6QhBdSc/FoG
bTpAnYIoZjCBE06/ZQkTjY2l0ZCsDjxrBPYUL9v20zffHBU0yV9BFtBSR+iKepoz
KZ4W7nCT5uk0di3EFyp8D5K2DMXdjzoi5hUHsV0sih8pJdTWhApq/6Cup/BBwUEA
n3/kD2rYvWY8J0I6RDhgCy31DbL7xG9eEPgJWFFBQxnDNQQtWclix8k0UT6gVM
QIuvqdz0boX7m8ZTx0ELJ0PgZTfxvviTx6dx3f1RxFADK50oLvBqNHcfAgMBAAGj
ggP8MIID+DAfBgNVHSMGDAWgBRQ6n0J2yn7EI+e5QEg1N55mUiD9zAdBgNVHQ4E
FgQU/if/AS6ZwpRYDvIeXapeipNHVg4wgccGA1UdEQSBvzCBvII0Ki5mYWNlYm9v
ay5jb22CDiouZmFjZWJvb2submV0gggqLmZiLmNvbYILKi5mYmNkbi5uZXSCCyou
ZmJzYnguY29tghAqLm0uZmFjZWJvb2suY29tgg8qLm1lc3Nlbmdlci5jb22CDiou
eHguZmJjZG4ubmV0gg4qLnh5LmZiY2RuLm5ldII0Ki54ei5mYmNkbi5uZXSCDGH
Y2Vib29rLmNvbYIGZmIuY29tgg1tZXNzZW5nZXIuY29tMA4GA1UdDwEB/wQEAWIF
```