Web servers

* What is a web server?

A web server is hardware and software that delivers contents to clients in the form of web pages. A web browser usually uses HTTP and hosts multiple web sites. A web server can support server side scripting and can be embedded into devices such as printers or routers.

* Client/server paradigm:

1. A server waits for requests to arrive.
2. Client sends requests for information
3. Server processes the request and sends back information
4. Clients can be other servers that need information
5. Client and server can be on separate machines, but can exist on the same hardware

* Stateful and stateless protocols:

1. Stateful

a) Maintains the state between specific commands

b) Server maintains the state between successive commands until the connection is terminated

c) Sequences of commands are grouped into a session.

d) One example of a stateful protocol is SMTP

1. Stateless
2. A single command with a single response
3. Doesn’t keep track of any previous requests.
4. Requests terminate, and the server forgets them.
5. One example is HTTP, the requests and responsive we’ve looked at in class

* Communicating using TCP/IP

1. The client’s program opens a socket, which is a connection between the client and server
2. The servers listen for requests on specific ports
3. Ports aren’t physical, just a numeric reference within a request that specifies which server program is supposed to get the request.
4. Three types of ports: Well known, registered and dynamic, or private.
5. SMTP – port 25
6. SSL – port 443
7. HTTP – ports 80

* Security and encryption

1. By default, data is always sent as plain text.
2. A packet sniffer comes in on the line to read the text
3. Secure Shell (SSH) is secure
4. 1024 or 2048 bit encryption is used
5. 2 keys are generated
6. A public key is generated for the host
7. A private key is generated by the client, which is unique
8. When the two talk, they exchange keys.

* Web requires only 3 components:

1. A markup language for formatting documents (XML or HTML)
2. A uniform notation for addressing resources on the network
3. A protocol for transporting messages over networks.

* Uniform Resource Locator – URLs:

1. Scheme://host:[port]/path/filename/[parameters][?query string][#anchor]
2. Scheme- protocol used (HTTP/HTTPS/FTP/etc.)
3. Host – the hostname or the host’s IP address
4. Port – The port the web server is listening on (optional)
5. Path – the file path on the web server to the desired document
6. File name – the name of the file on the web server
7. Parameters – URL parameters like the session Id
8. Query string – Name-value pairs passed to the document
9. Anchor – reference to specific position in the document

* Fundamentals of HTTP!

1. Foundation protocol of the web
2. Current version is HTTP/1.1
3. It’s used by clients and web servers to exchange information
4. Browsers used to only support HTTP and FTP, but now also include NNTP, mail, local files, etc
5. HTTP is an application level protocol in the TCP/IP suite which uses TCP as the transport layer. Going back to networking and talking about the different layers of a request/response.
6. Very simple protocol
7. Uses request/response paradigm
8. Is an example of a stateless protocol, so single request, single response.

* Structure of HTTP messages:

1. Requests and responses all have the same basic format:
2. Message header
3. **Blank line**
4. Message body
5. **Blank line**
6. Requests start with: METHOD /path-to-resource HTTP/version(1.1)
7. Responses start with: HTTP/version(1.1) status-code/explanation

* Request Methods:

1. GET
2. Simplest of the request methods
3. Does not have a body
4. Any parameters are sent as a query string directly in the URL
5. POST
6. Has a body containing the headers
7. Body contains parameters and form fields
8. GET and POST behave differently depending on the receiving application

* Status codes:

1 – informational

2 – successful response (200)

3 – Tells the client to redirect (300 – URL change)

4 – client request errors (404 – not found)

5 – server errors (500 – internal server error)

* Steps in processing a request:

1. Request arrives at server to networking support module
2. Request is passed to the address resolution module
3. After resolution and authentication, request is passed to request processing module
4. Control passes to the response generation module
5. Response is sent back to the networking support module
6. Response is sent back to the requestor

* Address resolution module:

1. Pre-processing done to the request
2. Virtual hosting: if the web server provides service for multiple domains, it determines which domain is being referenced.
3. Address mapping – figures out where to get the information on the server
4. Authentication – If resource is protected, checks credentials to make sure request is authorized

* Delivering static content

1. Static content page
2. Static files contain HTML pages, XML pages, plain text, images
3. HTTP responses must be constructed
4. As-is page
5. File containing complete HTTP responses including the headers

* Delivering dynamic content

1. CGI – Common Gateway Interface
2. First and most common for a long time
3. Spawns a separate process for each request received
4. Based on a fixed set of environment variables
5. Often implemented in Perl
6. Script is called as form action from web page
7. Data must be decoded by server application before processing
8. Quite a simple method

* Server security

1. Need to make servers secure, but not too much
2. Make sure you are only running the processes you need. If you aren’t using a port, close it. Stop running a protocol if it isn’t in use.
3. Only provide access needed.
4. If HTTP and FTP have the same access, then someone can use FTP to write a malicious executable to HTTP directory
5. Use SSL connection
6. Configure a firewall to only have certain ports open (do NOT run HTTP web server on the same machines as the firewall)
7. Use HTTP proxy with the firewall.
8. Protects the server behind the firewall
9. Does not processing, only routes requests to the right ports
10. Difficult to penetrate to where the real processing happens.

* Proxies

1. Requests made by a client are almost never directly to the server.
2. A proxy acts as both a client and a server
3. Makes requests on the behalf of other clients
4. Performs some security measures to protect the server, such as acting as a firewall.

AJAX- Asynchronous JavaScript and XML

* Refers to the combination of the Javascript and XML technologies.
* Allows web pages displayed on a client to quickly interact and exchange data with a web server without ever needing to reload the entire web page.
* AJAX primarily relies on JavaScript and HTTP requests to exchange data between a client and server
* XML is usually the format used for exchanging data

Problem:

* Getting information from a database on the server, or sending information to a server-side script like PHP,
* HTML form to GET or POST data to the server
* Click submit, wait for the server to respond
* Load a new page with results
* With AJAX
* Javascript communicates directly with server
* Uses a special JavaScript object called XMLHttpRequest
* JavaScript uses object to get information from the server without having to load new web page
* Other technologies that comprise AJAX
* HTML, CSS and the Document Object Model (DOM)

Introduction to AJAX

* XMLHttpRequest object:
* Uses http to exchange data between a client computer and a web server
* Can be used to request and receive data without reloading a web pag
* Combining XMLHttpRequest with the DOM
* You can update and modify individual portions of your web page with data received from the web server

Starting from the browser:

* Uses JavaScript to:
* Handle events in the form
* Create the XMLHttpRequest object
* Send it (via HTTP) to the server
* Receive the response
* Process the information returned from the server
* Nothing special is required of the server – every server can handle HTTP requests
* The XMLHttpRequest object does not actually require XML

The XMLHttpRequest object

* JavaScript has to create an XMLHttpRequest object before it can request items on the server
* Var request = new XMLHttpRequest();
* Other forms exist but they are for older browswers
* Use try/catch to make sure it works

XMLHttpRequest object methods

* Abort() :
* cancels the current HTTP request
* getAllResponseHeaders()
* returns a text string containing all of the response’s headers that were returned with a response in header : value, separated by line breaks
* getResponseHeader(header\_name)
* returns a text string containing the value assigned to the specified header
* open(method, URL, [,async,user,password])
* Specifies the method and URL for an HTTP request; assigning a value of true to the async argument performs the request asynchronously while a value of false performs the request synchronously; the default is true.
* Send ([content])
* Submits and HTTP request using the information assigned with the open() method; the optional content argument contains the message body
* setRequestHeader (header\_name, value)
* creates and HTTP request using the header\_name and value assigned

XMLHttpRequest object properties

* onreadystatechange
* specifies the name of the event