Faculty of Engineering and Applied Science

SOFE 4790U Distributed Systems

Homework #3

Name	Student #
Owen Musselman	100657709

Can you compile a list of software requirements of a web-server as described in the article?

- Parse the incoming requests to figure out what method is within the request.
- Create or find the page that the client is requesting.
- Send the page to the client that was requesting it.
- Check if the client requested page exists and send it to them if it exists.
- Need an error handler which sends the client errors if pages they request do not exist.
- Require a server exception handler which will signal if there are any internal errors in the server code.
- Need an IP and port so that it can be assigned a domain name.

At what layers of the OSI models does this system operate? What is the role of HTTP, DNS, HTML in the described system?

The OSI layers used in this system are 5, 4, 3, 2, 1. This is due to this system needing to communicate with the client which is done at layer 5, but in order for that to happen a session needs to be made which is done at this layer. Layer 5 will setup, coordinate and terminate the communication between the application involved. Layer 4 is used as it is responsible for for coordination of the data transfer between the server and the client, like where the client is, and at what rate the data needs to be transferred to the client. Layer 4 is where the TCP/IP protocol is. Layer 3 is used as it forwards the packers along a route. Layer 2 is used as it is responsible for error correction, along with node to the node data transfer. Layer 1 is used as it is the physical medium by which the data is transferred, where requests come in, and where responses go out.

The role of HTTP in this system is to request information using get requests. For the role of uploading information to the server, post requests are used. The HTTP url has the ability to have parameters included in it which provides even more information, like searches, and client browsers, etc. Additionally, HTTP is stateless, which means that each individual request is handled independently, with the server not storing anything about that request, if that information is needed it can be done in the form of cookies. Cookies are a short string of characters in which a server sends to a client. If the client performs actions that require the state of the client to be saved, a new cookie is made and then stored in a database, and then sent to the client. Any time the client reconnects the client sends the cookie to the server and the server checks the database for it and displays the page according to the saved state. HTTP is mainly used for data transfer over IP.

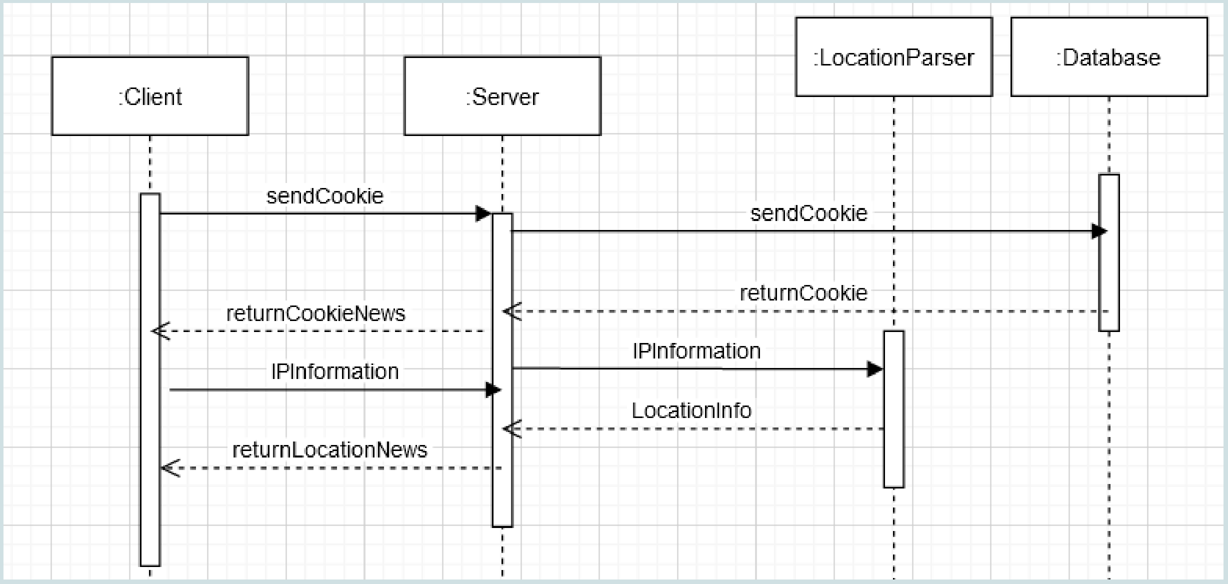
The role of DNS in this system is to just assign the server's 8-bit IP address to a name that is chosen by the server owner so that the address to the server's site can be easily remembered by clients. For example an IP of 24.224.35.149 is assigned to foobar.com and instead of people remembering 24.224.35.149, they can just remember foobar.com.

The role of HTML is to display the generated and the retrieved page that the client had requested from the server to the client's browser. Headers are used to tell the client to interpret the received data from the server as HTML.

Take CNN.Com for example, their webserver updates the information displayed on each client? How can a web server do this? Draw UML sequence diagrams? What functions/ components need to be added to the described system?

Using cnn.com as an example, their web server updates the information that is displayed on each client's browser by possibly getting their general location by their IP address, and displaying news that is more applicable to that location. Another their web server can display more personalized information by using cookies, and displaying news that is similar to what searches have been made, or previous visits to the cnn site.

UML sequence diagram of CNN web server



The functions/components added are the location parser and the database. Where the database holds the cookie information of the clients, where it is used to validate the clients accessing the site with previous sessions. The web server will present the client with a page that reflects their previous session information. The location parser is used to get the general location of the client as to have a location based web page presented to the client upon access.

How are transparencies provisioned within the described image sharing system?

The transparencies provisioned within this system is that the client has no idea how the requests are being sent to the server, or how the responses are being sent. That information is hidden from the client's view. Another transparency is the Error handling, where the client only sees the status messages: informational responses (100-199), successful responses (200-299), redirect responses (300-399), client error responses (400-499), server error responses (500-599). Additionally the client is presented with the responses information like http version, header field names they are only presented with the body which is usually html. When the client gets a response with html it is interpreted by the browser and displays it accordingly, it doesn't just display it as html code.

The following transparencies are provisioned within the image-sharing web-server:

- Access Transparency: The data of many images stored on multiple servers are hidden from the client.
- Location Transparency: The data returned to the user, such as an image or a document, will be displayed, but no information regarding the physical location of the file will be accessed by the user.
- Migration Transparency: The image being moved or transferred onto a new physical server, or the directory of the image being updated within the server is changed. The user should not be impacted when they are accessing the website.

What is ' synchronous messaging" or request reply messaging? what is the role of a web server in this type of messaging? How is that different from Asynchronous Messaging Primer?

Synchronous messaging/request reply messaging will block the execution of code in the browser, which inherently pauses the user experience until a response is received. This is done when the client requests to visit a page on a web server, where the client requests the page, the client will then wait for the request to be served. The server will respond to the request with the requested page. The client then gets the page it requested and the user experience is resumed. Asynchronous messaging is different as there is an intermediate messaging broker which provides functionality of moving messages from the producer to the consumer. The broker can assist in async messaging processing when there are operations being executed that take longer to complete, so if a command is sent, the producer will not have to wait for the consumer to complete with the inclusion of the broker.

What is the role of a web server in Queue-Based Load Leveling Pattern

The role of the web server in the queue-based load leveling pattern is that the web server will direct the various tasks to a service bus queue, which the queue then decouples the tasks from the services, so that the services are able to handle messages now, even if there are a large number of requests coming from the web server.

What can you do with CGI ? what are the pros and cons? How can we enhance CGI? Read about three-tier architecture and multi-tier architectures ... How did this functionality evolved from CGI?

CGI is used to grant a way for a web server to execute programs externally to satisfy a request. Some pros of CGIs are: give a simple interface for creating HTTP requests to a web server. You do not need special libraries/APIs to make a CGI. CGIs can be written in many languages, support for CGIs is very extensive. Some cons of CGIs: Lots of processing time taken. HTTP requests to the server can become slow. It is harder to cache data in memory when going between pages. CGI can be enhanced through the use of plugins like FastCGI where the plugin enables a web server to safely work in conjunction with content generating technologies like python and perl. We can also enhance CGI to be more secure so even if a user knows the path to a python file stored on the server they cannot execute it, or only some sections of it. multi -tier architectures evolved from CGI as, like the name suggests there are multiple layers in an architecture, and the CGI is the middle man interface which web servers pass the client requests to the databases and other resources. In essence CGI is implemented through the business logic layer as it acts as the middleman to communicate with the external resources.

A web server is just one piece of the "enterprise architecture" what are other pieces? How can such architecture handle very large amounts of data? Where would you fit a Database? How can the operating system support such a webserver?

There are various parts of an enterprise system, with there being security, application, infrastructure, information, networks, integration. The web server fits into the infrastructure portion, with the other components of a system falling into other sections. Enterprise architecture can handle large amounts of data due to the ability to utilize the cloud for containerization, and virtualization, which allows for load balancing, request splitting, and resource scalability and elasticity. The architecture can handle large amounts of data through the use of a bottom up approach where the raw data is stored in the system and recording the metadata for the website analytics. It uses the information architecture that will always have real time data sources that are capturing messages in real time.

The other pieces of the enterprise architecture is the business architecture. Next it drives down the information architecture that prescribes itself onto information systems architecture.

Next it translates its data to data architecture, which is supported by the delivery systems architecture that all work cohesively with the web server.

Using different database models, and individualizing them to use different data sources. For example, having a model expose itself to an API template or to a web server, would allow us to fit the information and spread it across the database as required.

A web server is usually installed on the server Operating system, which allows the user to deploy or install business layer and web applications.