Laboratorio #02 - Virtual

Protocolo HTTP

Ciencias de la Computación VIII

Instrucciones:

Este laboratorio tiene como objetivo entender la estructura los **encabezados** de request y response de **HTTP**, además realizar la lectura del **RFC 1945, RFC 2616, RFC 7540 y QUIC** de sus versiones, además de complementarios **RFC 5321 y RFC 959.**

HTTP significa **Hypertext Transfer Protocol**. Toda la World Wide Web utiliza este protocolo, establecido a principios de los 90. Casi todo lo que aparece en un navegador ha sido transmitido a través de HTTP mediante requests y responses entre navegador y servidor. Actualmente se recomienda utilizar **HTTPS** para evitar enviar un texto plano por la red y que esté sea cifrado de punto a punto.

Los **HTTP headers** son la parte central de los HTTP **requests** y **responses** y transmiten información acerca del navegador del cliente de la página solicitada, del servidor, etc.

Cada vez que se visita un sitio, puedes ver los headers del request en la consola del navegador:

Ejemplo:

Request Header	**Response Header**
GET /test.html HTTP/1.1 Host: www.galileo.edu User-Agent: Mozilla/5.0 Accept-Language: es,en Connection: keep-alive	HTTP/1.1 200 OK Server: MiServer/1.2.3 Date: Wed, 20 Dec 1989 7:07:01 GMT Content-Type: text/html Connection: keep-alive Content-Length: 29 <HTML>Test: Hola Mundo</HTML>

Estructura de un HTTP request:
Método: GET, POST, PUT, etc. Indica que tipo de request es.
Path: la URL que se solicita.
Protocolo: contiene HTTP y su versión.

Ejemplo: GET /test.html HTTP/1.1

Headers: Esquemas de key:value que contienen información sobre el HTTP request y el navegador.
Body: Si se envía información al servidor a través de POST o PUT, ésta va en el body.

Estructura de un HTTP response:
El navegador envía el **HTTP request**, el servidor responde con un **HTTP response**, compuesto por:

Protocolo: Contiene HTTP y su versión, actualmente 1.1.
Status Code: El código de respuesta, por ejemplo: 200 OK, que significa que el GET request ha sido satisfactorio y el servidor devolverá los contenidos del documento solicitado. Otro ejemplo es 404 Not Found, el servidor no ha encontrado el resource solicitado.

Ejemplo: HTTP/1.1 200 OK

Headers: Contienen información sobre el software del servidor, cuando se modificó por última vez el resource solicitado, el mime type, etc. La mayoría son opcionales.
Body: Si el servidor devuelve información que no sean headers ésta va en el body.

Descripción:
Debe de implementar un servidor **Multithreading** en **PERL**, de manera tal que pueda atender a múltiples requests a la vez, en el **puerto 8055.**

Implementar los métodos de petición **HEAD, GET y POST** (**RFC HTTP - sección 5.1.1**), luego de analizar el encabezado del request, su servidor debe procesar la información y construir el HTTP **response** acorde a lo que se solicita (**RFC HTTP - sección 6**), si se solicita con otro método no soportado, no se encuentra el objeto solicitado o existe otro tipo de razón, responder con el archivo, formato y status-code correspondiente.

En el ejemplo se solicita test.html al servidor, si el objeto existe, debe construir el **HTTP response** con **status-code 200 OK** y conteniendo el objeto solicitado y enviarlo como respuesta al cliente. Si el objeto no existe, debe construir el **HTTP response** correspondiente y crear una **status-page con la razón** y enviarlo como respuesta al cliente; esto en el caso del GET.
Recuerde que si no se indica **objeto**, por defecto es **index.html**

Pruebas:
Durante el desarrollo utilice **Telnet** para realizar los request utilizando los métodos **HEAD, GET o POST**, las pruebas finales las puede hacer desde el web browser para que sea interpretado el HTML. Tome en cuenta que el URL en el browser sería: http://localhost:8055 o http://127.0.0.1:8055, si no coloca :8055 su browser intentará abrir una conexión al puerto por default, en este caso al puerto 80.
Durante su desarrollo verifique que su servidor funcione con ngrok para que se pueda conectar alguien más a su servidor.

Entrega:
- Se le adjunta un código base con inicio del laboratorio.
- Debe entregar solo el archivo **PERL** llamado **TCPServer.pl** con su solución de los métodos **HEAD, GET** y **POST.**
- El laboratorio debe ser entregado por medio del GES en un **ZIP.**
- El laboratorio puede tener una calificación de cero si no compila o se detecta plagio.