Universidad de Valparaíso CHILE

REDES DE COMPUTADORES: TAREA 2

Implementación de un servidor web TCP en C++

Entendimiento modelo 4+1

Nombres:

Nicolas Ignacio Toledo Toledo
Leonardo Esteban Arancibia Vicencio
Augusto Ignacio Pinochet Contreras

Profesor:

Gabriel Astudillo

13/12/2018
REDES DE COMPUTADORES I

INTRODUCCIÓN

Hoy en día, en nuestros dispositivos hacemos miles de conexiones por minuto, claro, si somos usuarios excesivos de Internet. Lo que se nos pidió en esta tarea fue replicar una de esas conexiones creando nuestro propio servidor iterativo simple, con el fin de retornar al navegador web una pagina HTML simple, no muy extensa, mientras el servidor se está ejecutando por consola, generando así al grupo un desafío por superar, ya que este proyecto depende del primero.

OBJETIVOS

- Implementar Servidor web TCP Simple en C++14.
- Conocer y aplicar el API de Networkning en C++.
- Implementar páginas web sencillas, sólo texto HTML
- Manejo de intercambio de datos
- Manejo de archivos JSON
- Manejo de SOCKETS
- Manejo del lenguaje C++14
- Manejo del lenguaje HTML
- Manejo de procesos paralelos

DESARROLLO

Para comenzar nos encontramos con el problema de que no sabíamos como implementar lo aprendido en la tarea anterior con lo relacionado de ahora. Investigando un poco, encontramos el request que da un navegador web, el cual se compone de muchas líneas, en la primera se encuentra el request del archivo HTML. Desarrollando un poco esa idea, la metodología que ocupamos fue comparar el echoBuffer que entrega el socket, buscando en la primera línea del request el HTML requerido y compararlo con el nombre de los archivos HTML que tenemos creados en su directorio correspondiente (“www-data”, “www-error”).

Las modificaciones al código entregado no fueron tantas, el cambio mas radicar fue el eliminar un ciclo While por completo, que para el funcionamiento de lo que queríamos hacer, no venían al caso.

Se adjuntan fotos del código ya modificado con sus respectivos comentarios.

Código final

Presentaremos capturas de pantalla del código final para posteriormente explicar cuáles fueron los cambios realizados al código fuente

```
51  //SE ENVIA A TRAVÉS DEL SOCKET EL RESPONSE AL REQUEST INICIAL
52  sock->send("HTTP/1.1 200 OK\r\n", 32);//HTTP indicado en la especificacion de la tarea
53  sock->send("Content-Type: text/html\r\n\r\n", 32);
54  //SE GENERAN LAS VARIABLES NECESARIAS PARA COMPRAR EL ECHOBUFFER CON EL NOMBRE DE LOS ARCHIVOS
55  //ADEMAS SE GENERAN VARIABLES PARA PODER LEER LINEA POR LINEA EL ARCHIVO HTML
56  std::string leerlinea;
57  std::string htmlstruct="";
58  std::string rqe(echoBuffer, 5, 6);
59  std::string a1 = "1.html";
60  std::string a2 = "2.html";
61  std::string a3 = "3.html";
62  std::ifstream arch("../www-data/1.html");//Tola aqui se coloca la pagina1 data(" ")
63  std::ifstream arch1("../www-data/2.html");
64  std::ifstream arch2("../www-data/3.html");
65  std::ifstream arch404("../www-error/404.html");
66  std::cout << rqe + "\n";
67  //SI EL SUBSTRING rqe PROVENIENTE DEL ECHOBUFFER ES IGUAL AL NOMBRE DEL ARCHIVO 1
68  //SE LEE LINEA POR LINEA EL HTML Y LO ENVIA A TRAVÉS DEL SOCKET
69  //ESTA MISMA CONDICION SE REPLICA PARA EL ARCHIVO 2, 3 Y EL ARCHIVO 404
70  //ESTE ULTIMO SOLO SE VA A EJECUTAR SI EL REQUES NO CORRESPONDE A NINGUN NUMBRE DE
71  //NINGUN ARCHIVO.
72  if(rqe == a1){
73    if (arch.is_open()){
74      while(getline(arch, leerlinea)){
75        htmlstruct = htmlstruct + leerlinea + "\n";
76      }
77      arch.close();
78      sock->send(htmlstruct.c_str(), htmlstruct.length());
79    }
80    else{
81      std::cout << "No se ha podido abrir el HTML correspondiente";
82    }
83  }
84  else if (rqe == a2){
85    if (arch1.is_open()){
86      while(getline(arch1, leerlinea)){
87        htmlstruct = htmlstruct + leerlinea + "\n";
88      }
89      arch1.close();
```

```
90        sock->send(htmlstruct.c_str(), htmlstruct.length());
91      }
92      else{
93        std::cout << "No se ha podido abrir el HTML correspondiente";
94      }
95    }
96    else if (rqe == a3){
97      if (arch2.is_open()){
98        while(getline(arch2, leerlinea)){
99          htmlstruct = htmlstruct + leerlinea + "\n";
100       }
101       arch2.close();
102       sock->send(htmlstruct.c_str(), htmlstruct.length());
103     }
104     else{
105       std::cout << "No se ha podido abrir el html correspondiente";
106     }
107   }
108   else{
109     if(arch404.is_open()){
110       while(getline(arch404, leerlinea)){
111         htmlstruct = htmlstruct + leerlinea + "\n";
112       }
113       arch404.close();
114       sock->send(htmlstruct.c_str(), htmlstruct.length());
115     }
116
117     else{
118       std::cout << "No se ha podido abrir el html correspondiente";
119     }
120   }
121   delete sock;
122 }
```

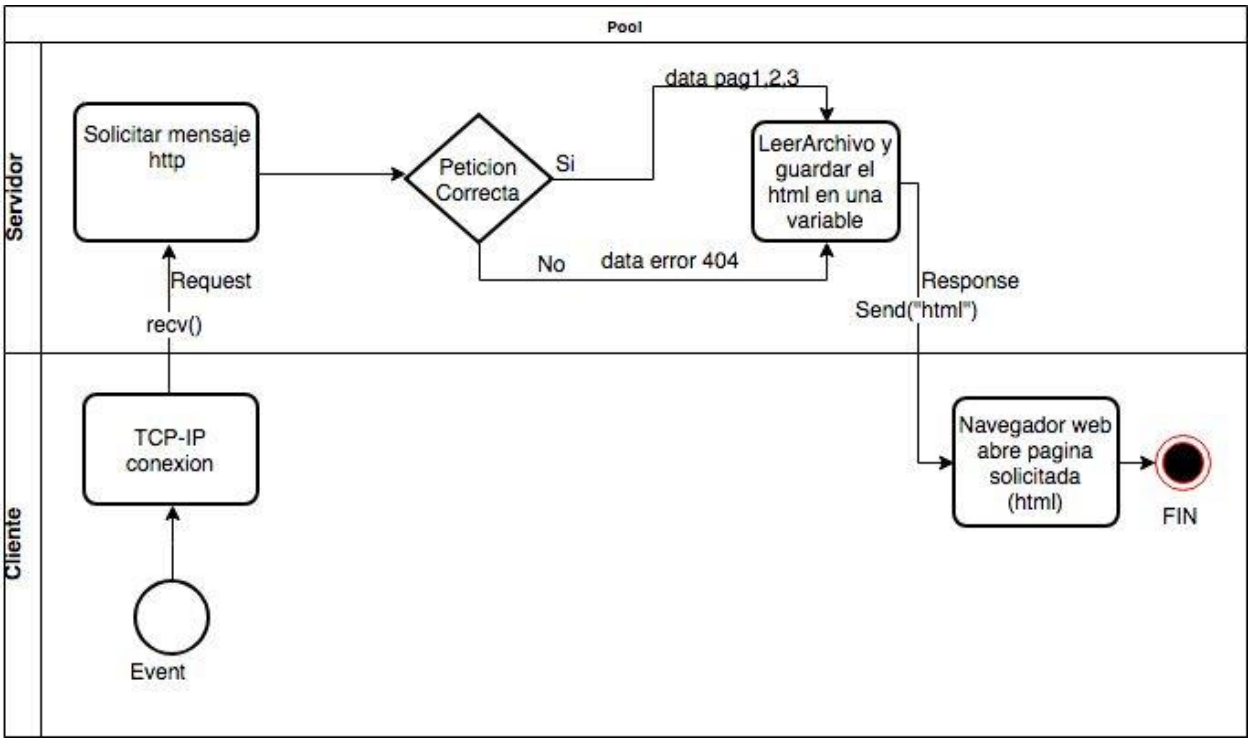
Al final lo que se desarrollo mas en el código, fue la comparación de strings que permitían saber si la página requerida existía en los directorios que definimos en su respectiva sección de código. Una vez comparados ambos strings, se enviaba a través del socket, línea por línea, para mostrarlo en el navegador web, en este caso, como era una pagina HTML simple, el trabajo requerido para recorrer línea por línea el archivo HTML, no es tan pesado.

Lo demás, como variables, ciclos, están comentados en el código en sí, para no hacer tan extenso el funcionamiento del código y no entrar en divagaciones.

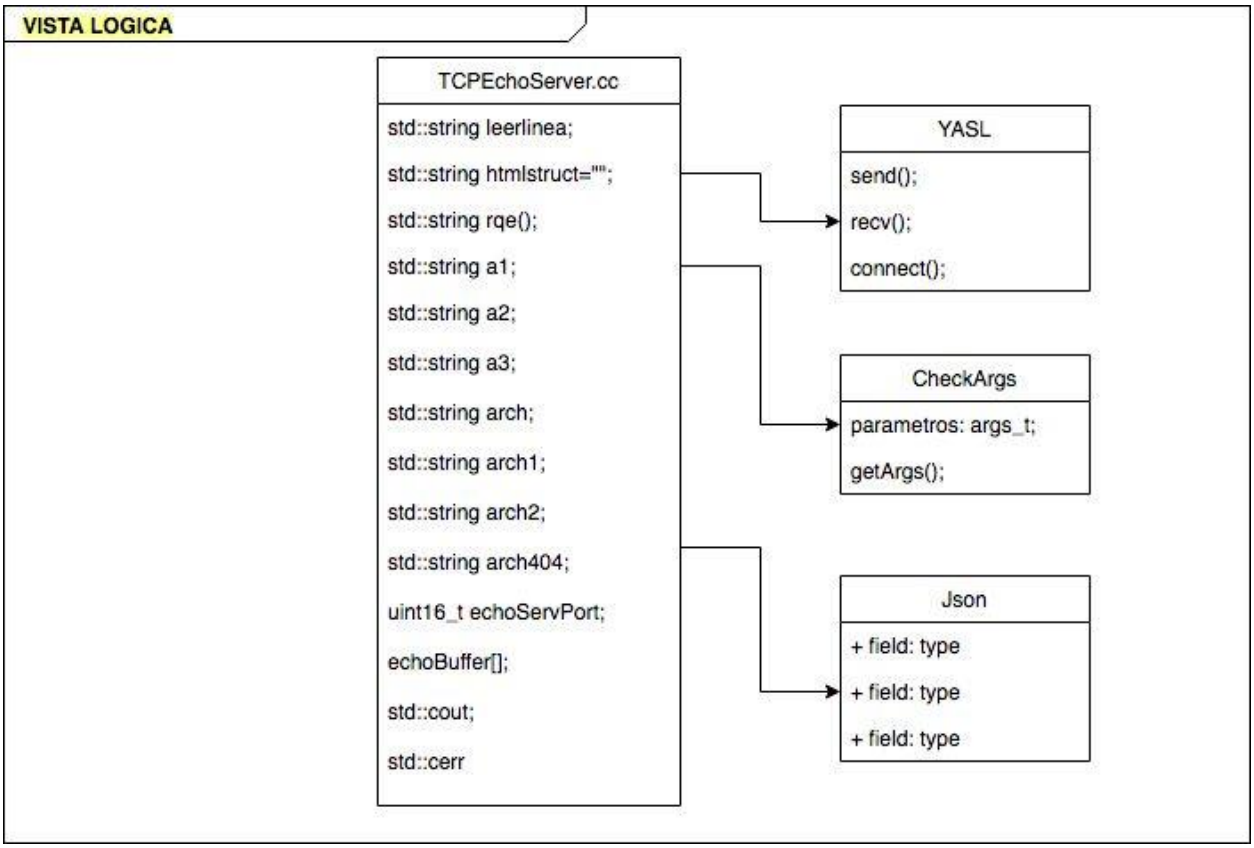
Arquitectura de software

Representamos las arquitecturas del software a través del modelo 4+1 en donde tomaremos 2 “Vistas” de este modelo las cuales serán: la Vista Lógica y la Vista Proceso.

Vista Proceso



Vista lógica



CONCLUSIÓN

El mayor desafío de este proyecto para el grupo fue poder entender como funcionaban en si los servidores web, para así poder replicar su funcionamiento de una manera mas sencilla y solo con paginas HTML simples. Como se planteó en el proyecto anterior, la manera de programar en función a objetos que están rondando por la red, genera un pensamiento de programación distinto al que hemos desarrollado hasta ahora, de solo programas “offline”. Lo que queda pendiente como grupo, es aprender un poco mejor el funcionamiento de los archivos JSON y para que sirven, ya que dentro de nuestra ignorancia no pudimos hacer efectiva la utilización de estos tipos de archivos, tal es el caso de la variable root_dir o notFoundFile ya que estas no fueron utilizadas.

LINKOGRAFÍA

http://www.cplusplus.com/reference/stl

https://www.boost.org