

REDES DE COMPUTADORES: TAREA 1

Implementación de un cliente HTTP en C++

Entendimiento modelo 4+1

Nombres:

Nicolas Ignacio Toledo Toledo

Leonardo Esteban Arancibia Vicencio

Augusto Ignacio Pinochet Contreras

Profesor:

Gabriel Astudillo

15/11/2018

REDES DE COMPUTADORES I

INTRODUCCIÓN

Existen varios tipos de protocolos que se desarrollan a lo largo de la carga de una página web, algunos son vistos por el usuario y otros no, los primeros son claramente los objetos visualizados en la pantalla del computador, y los otros simplemente son complejos de entender. El propósito de este trabajo es investigar cuántos de esos protocolos (vistos y no vistos) son ejecutados al poder navegar por el internet, además, ampliar nuestros conocimientos en los lenguajes de programación y nuestras mentes como ingenieros, desarrollando un programa en el lenguaje C++, que permita poder guardar una página web en un archivo “txt”, todo esto realizándose desde consola.

OBJETIVOS

- Implementar cliente HTTP simple en C++14.
- Capturar el tráfico HTTP a través de wireshark hacía muchos servidores web
- Poder entender qué es HTTP y HTML, y cómo están relacionados
- Analizar cuales son las acciones y protocolos HTTP que se utilizan en la conexión
- Implementación del cliente HTTP basado en texto
- Guardar código HTML de una página web en un archivo de texto
- Poder entender de qué trata el modelo 4+1 e implementarlo en el proyecto

DESARROLLO

¿Que es HTTP?:

El Protocolo de transferencia de hipertexto, es el mecanismo primario para transportar objetos por la web y es una aplicación de protocolo (define cómo interactúan un cliente y un servidor.)

HTTP es un protocolo de [petición-respuesta], el cual significa que un cliente envía un mensaje de petición y el servidor retorna un mensaje de respuesta (el mensaje está basado en texto entendible para humanos.)

HTTP también es llamado un protocolo sin estado, a lo cual se refiere a que es un protocolo de comunicaciones que trata cada petición como una transacción independiente que no tiene relación con las solicitudes anteriores.

¿Qué es HTML?:

HyperText Markup Language(Lenguaje de Marcado para Hipertextos) es la representación estándar para documentos de hipertextos, HTML define el diseño y el formato de una página web y permite a los autores incrustar referencias de hipervínculo a otros recursos de la web.

¿Cuál es la relación de HTML con HTTP?:

La relación existente entre http y html es que el protocolo de transferencia de hipertexto(http) sirve para transferir páginas de formato de hipertexto(html) desde un servidor web a un navegador web, para intercambiar páginas web entre servidor y buscador.(cliente-servidor, petición-respuesta).

Esquemas de páginas

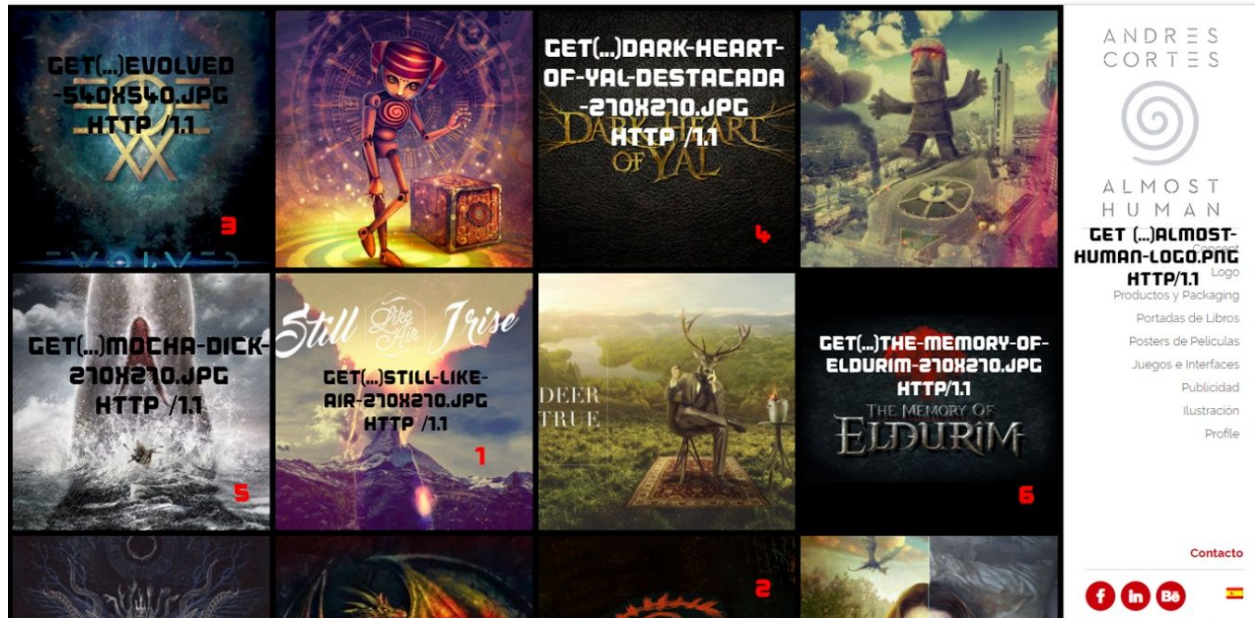
Página 1

NOMBRE DE LA PÁGINA: ALMOSTHUMAN.

TOTAL PROTOCOLOS HTTP: 82.

URL DE LA PÁGINA: ALMOSHUMAN.CL

OBJETOS QUE TIENE: IMAGENES, LOGO DE LA PÁGINA E HIPERVINCULOS A SUBPÁGINAS



Sniffee página 1

No.	Time	Source	Destination	Protocol	Length	Info
1423	17.541219	192.168.0.15	50.87.144.211	TCP	54	50137 → 80 [ACK] Seq=1019 Ack=202540 Win=470016 Len=0
1421	17.541206	50.87.144.211	192.168.0.15	TCP	1514	80 → 50137 [ACK] Seq=279814 Ack=1019 Win=31360 Len=1460 [TCP segment of a reassembled PDU]
1420	17.541205	50.87.144.211	192.168.0.15	TCP	1514	80 → 50137 [ACK] Seq=278354 Ack=1019 Win=31360 Len=1460 [TCP segment of a reassembled PDU]
1419	17.540807	192.168.0.15	50.87.144.211	TCP	54	50137 → 80 [ACK] Seq=1019 Ack=278354 Win=470016 Len=0
1418	17.540795	50.87.144.211	192.168.0.15	TCP	1514	80 → 50137 [ACK] Seq=276894 Ack=1019 Win=31360 Len=1460 [TCP segment of a reassembled PDU]
1417	17.540795	50.87.144.211	192.168.0.15	TCP	1514	80 → 50137 [ACK] Seq=275434 Ack=1019 Win=31360 Len=1460 [TCP segment of a reassembled PDU]
1416	17.540793	50.87.144.211	192.168.0.15	TCP	1514	80 → 50137 [ACK] Seq=273974 Ack=1019 Win=31360 Len=1460 [TCP segment of a reassembled PDU]
1415	17.540793	50.87.144.211	192.168.0.15	TCP	1514	80 → 50137 [ACK] Seq=272514 Ack=1019 Win=31360 Len=1460 [TCP segment of a reassembled PDU]
1414	17.540587	192.168.0.15	50.87.144.211	TCP	54	50137 → 80 [ACK] Seq=1019 Ack=272514 Win=470016 Len=0
1413	17.540573	50.87.144.211	192.168.0.15	TCP	1514	80 → 50137 [ACK] Seq=271054 Ack=1019 Win=31360 Len=1460 [TCP segment of a reassembled PDU]
1412	17.540572	50.87.144.211	192.168.0.15	TCP	1514	80 → 50137 [ACK] Seq=269594 Ack=1019 Win=31360 Len=1460 [TCP segment of a reassembled PDU]
1411	17.540572	50.87.144.211	192.168.0.15	TCP	1514	80 → 50137 [ACK] Seq=268134 Ack=1019 Win=31360 Len=1460 [TCP segment of a reassembled PDU]
1410	17.540571	50.87.144.211	192.168.0.15	TCP	1514	80 → 50137 [ACK] Seq=266674 Ack=1019 Win=31360 Len=1460 [TCP segment of a reassembled PDU]
1409	17.540570	50.87.144.211	192.168.0.15	TCP	1514	80 → 50137 [ACK] Seq=265214 Ack=1019 Win=31360 Len=1460 [TCP segment of a reassembled PDU]
1408	17.540359	50.87.144.211	192.168.0.15	TCP	1514	80 → 50137 [ACK] Seq=263754 Ack=1019 Win=31360 Len=1460 [TCP segment of a reassembled PDU]
1407	17.540156	192.168.0.15	50.87.144.211	TCP	54	50137 → 80 [ACK] Seq=1019 Ack=263754 Win=470016 Len=0
1406	17.540143	50.87.144.211	192.168.0.15	TCP	1514	80 → 50137 [ACK] Seq=262294 Ack=1019 Win=31360 Len=1460 [TCP segment of a reassembled PDU]
1405	17.540142	50.87.144.211	192.168.0.15	TCP	1514	80 → 50137 [ACK] Seq=260834 Ack=1019 Win=31360 Len=1460 [TCP segment of a reassembled PDU]
1404	17.540142	50.87.144.211	192.168.0.15	TCP	1514	80 → 50137 [ACK] Seq=259374 Ack=1019 Win=31360 Len=1460 [TCP segment of a reassembled PDU]
1403	17.540140	50.87.144.211	192.168.0.15	TCP	1514	80 → 50137 [ACK] Seq=257914 Ack=1019 Win=31360 Len=1460 [TCP segment of a reassembled PDU]
1402	17.540140	50.87.144.211	192.168.0.15	TCP	1514	80 → 50137 [ACK] Seq=256454 Ack=1019 Win=31360 Len=1460 [TCP segment of a reassembled PDU]
1401	17.539934	192.168.0.15	50.87.144.211	TCP	54	50137 → 80 [ACK] Seq=1019 Ack=256454 Win=470016 Len=0
1400	17.539921	50.87.144.211	192.168.0.15	TCP	1514	80 → 50137 [ACK] Seq=254994 Ack=1019 Win=31360 Len=1460 [TCP segment of a reassembled PDU]
1399	17.539920	50.87.144.211	192.168.0.15	TCP	1514	80 → 50137 [ACK] Seq=253534 Ack=1019 Win=31360 Len=1460 [TCP segment of a reassembled PDU]
1398	17.539920	50.87.144.211	192.168.0.15	TCP	1514	80 → 50137 [ACK] Seq=252074 Ack=1019 Win=31360 Len=1460 [TCP segment of a reassembled PDU]
1397	17.539918	50.87.144.211	192.168.0.15	TCP	1514	80 → 50137 [ACK] Seq=250614 Ack=1019 Win=31360 Len=1460 [TCP segment of a reassembled PDU]
1396	17.539712	50.87.144.211	192.168.0.15	TCP	1514	80 → 50137 [ACK] Seq=249154 Ack=1019 Win=31360 Len=1460 [TCP segment of a reassembled PDU]
1395	17.539505	192.168.0.15	50.87.144.211	TCP	54	50137 → 80 [ACK] Seq=1019 Ack=249154 Win=470016 Len=0
1394	17.539493	50.87.144.211	192.168.0.15	TCP	1514	80 → 50137 [ACK] Seq=247694 Ack=1019 Win=31360 Len=1460 [TCP segment of a reassembled PDU]
1393	17.539491	50.87.144.211	192.168.0.15	TCP	1514	80 → 50137 [ACK] Seq=246234 Ack=1019 Win=31360 Len=1460 [TCP segment of a reassembled PDU]
1392	17.539490	50.87.144.211	192.168.0.15	TCP	1514	80 → 50137 [ACK] Seq=244774 Ack=1019 Win=31360 Len=1460 [TCP segment of a reassembled PDU]

> Frame 221: 54 bytes on wire (432 bits), 54 bytes captured (432 bits) on interface 0
 > Ethernet II, Src: AsustekC_84:5a:55 (b0:6e:bf:84:5a:55), Dst: ArrisGro_e4:95:08 (c8:05:c2:e4:95:08)
 > Internet Protocol Version 4, Src: 192.168.0.15, Dst: 50.87.144.211
 > Transmission Control Protocol, Src Port: 50109, Dst Port: 80, Seq: 1, Ack: 1, Len: 0

0000 c8 05 c2 e4 95 08 b0 6e bf 84 5a 55 00 00 45 00n..ZU...E..

Ethernet: <live capture in progress> Packets: 3469 - Displayed: 444 (12.8%) Profile: Default

Página 2

NOMBRE DE LA PÁGINA: LECTURALIA.COM

URL DE LA PÁGINA: HTTP://WWW.LECTURALIA.COM

OBJETOS QUE TIENE LA PÁGINA: HIPERVINCULOS HACIA SUBPÁGINAS
(SRC DEL HIPERVINCULO SIN PETICION HTTP)

TOTAL PROTOCOLOS HTTP: 2

GET / HTTP/1.1
HTTP/1.1 200 OK
(TEXT/HTML)

Lecturalia Autores Libros Premios Blog Comunidad

Red social de literatura, comunidad de lectores y comentarios de libros

98.202 libros, 22.378 autores y 78.803 usuarios registrados

Autor destacado



Santiago Posteguillo
Santiago Posteguillo, uno de los autores de moda para los amantes de la novela histórica se ha alzado con el Premio Planeta en 2018 con Yo, Julia.

Libro destacado



Yo, Julia
Santiago Posteguillo
La novela histórica Yo, Julia, de Santiago Posteguillo, es la ganadora del Premio Planeta 2018.

Libros

- > Libros actualizados
- > Novedades editoriales
- > Libros mejor valorados
- > Libros más visitados
- > Últimos comentarios
- > Nuevos libros
- > Libros más destacados

Blog

Dear Holmes, la web para los detectives victorianos más audaces

6 de noviembre

- > ¿Serás capaz de resolver un misterio digno de Holmes?
- > Se trata de una interesante propuesta epistolar.

Sherlock Holmes es el detective de ficción más conocido de todos los tiempos. Se han adaptado sus aventuras al cine, la televisión, el teatro y los videojuegos, y todavía hoy siguen apareciendo ideas novedosas relacionadas con la gran creación de Arthur Conan Doyle. Un ejemplo que mezcla a la vez...

Leer más

Autores

- > Últimos actualizados
- > Autores más visitados
- > Autores destacados
- > Nuevos autores

Temas

- > Literatura

Sniffeeo página 2

No.	Time	Source	Destination	Protocol	Length	Info
194	2.872246	192.168.0.15	93.93.64.123	HTTP	697	GET / HTTP/1.1
312	3.490843	93.93.64.123	192.168.0.15	HTTP	195	HTTP/1.1 200 OK (text/html)
147	2.502936	192.168.0.15	93.93.64.123	TCP	66	50206 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1
148	2.503043	192.168.0.15	93.93.64.123	TCP	66	50207 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1
157	2.569009	192.168.0.15	93.93.64.123	TCP	66	50210 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1
169	2.695177	192.168.0.15	93.93.64.123	TCP	66	50212 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1
183	2.819953	93.93.64.123	192.168.0.15	TCP	66	80 → 50207 [SYN, ACK] Seq=0 Ack=1 Win=29200 Len=0 MSS=1460 SACK_PERM=1 WS=256
184	2.820017	192.168.0.15	93.93.64.123	TCP	54	50207 → 80 [ACK] Seq=1 Ack=1 Win=65536 Len=0
186	2.847039	93.93.64.123	192.168.0.15	TCP	66	80 → 50206 [SYN, ACK] Seq=0 Ack=1 Win=29200 Len=0 MSS=1460 SACK_PERM=1 WS=256
187	2.847089	192.168.0.15	93.93.64.123	TCP	54	50206 → 80 [ACK] Seq=1 Ack=1 Win=65536 Len=0
192	2.872055	93.93.64.123	192.168.0.15	TCP	66	80 → 50210 [SYN, ACK] Seq=0 Ack=1 Win=29200 Len=0 MSS=1460 SACK_PERM=1 WS=256
193	2.872112	192.168.0.15	93.93.64.123	TCP	54	50210 → 80 [ACK] Seq=1 Ack=1 Win=65536 Len=0
202	2.959921	93.93.64.123	192.168.0.15	TCP	66	80 → 50212 [SYN, ACK] Seq=0 Ack=1 Win=29200 Len=0 MSS=1460 SACK_PERM=1 WS=256
203	2.959970	192.168.0.15	93.93.64.123	TCP	54	50212 → 80 [ACK] Seq=1 Ack=1 Win=65536 Len=0
237	3.176634	93.93.64.123	192.168.0.15	TCP	60	80 → 50210 [ACK] Seq=1 Ack=644 Win=30720 Len=0
238	3.177859	93.93.64.123	192.168.0.15	TCP	1514	80 → 50210 [ACK] Seq=1 Ack=644 Win=30720 Len=1460 [TCP segment of a reassembled PDU]
239	3.179343	93.93.64.123	192.168.0.15	TCP	1514	80 → 50210 [ACK] Seq=1461 Ack=644 Win=30720 Len=1460 [TCP segment of a reassembled PDU]
240	3.179368	192.168.0.15	93.93.64.123	TCP	54	50210 → 80 [ACK] Seq=644 Ack=2921 Win=65536 Len=0
241	3.179738	93.93.64.123	192.168.0.15	TCP	1514	80 → 50210 [ACK] Seq=2921 Ack=644 Win=30720 Len=1460 [TCP segment of a reassembled PDU]
242	3.180017	93.93.64.123	192.168.0.15	TCP	1514	80 → 50210 [ACK] Seq=4381 Ack=644 Win=30720 Len=1460 [TCP segment of a reassembled PDU]
243	3.180028	192.168.0.15	93.93.64.123	TCP	54	50210 → 80 [ACK] Seq=644 Ack=5841 Win=65536 Len=0
244	3.180251	93.93.64.123	192.168.0.15	TCP	1514	80 → 50210 [ACK] Seq=5841 Ack=644 Win=30720 Len=1460 [TCP segment of a reassembled PDU]
245	3.180253	93.93.64.123	192.168.0.15	TCP	1514	80 → 50210 [ACK] Seq=7301 Ack=644 Win=30720 Len=1460 [TCP segment of a reassembled PDU]
246	3.180253	93.93.64.123	192.168.0.15	TCP	1514	80 → 50210 [ACK] Seq=8761 Ack=644 Win=30720 Len=1460 [TCP segment of a reassembled PDU]
247	3.180268	192.168.0.15	93.93.64.123	TCP	54	50210 → 80 [ACK] Seq=644 Ack=10221 Win=65536 Len=0
248	3.180490	93.93.64.123	192.168.0.15	TCP	1514	80 → 50210 [ACK] Seq=10221 Ack=644 Win=30720 Len=1460 [TCP segment of a reassembled PDU]
249	3.180490	93.93.64.123	192.168.0.15	TCP	1514	80 → 50210 [ACK] Seq=11681 Ack=644 Win=30720 Len=1460 [TCP segment of a reassembled PDU]
250	3.180492	93.93.64.123	192.168.0.15	TCP	1514	80 → 50210 [ACK] Seq=13141 Ack=644 Win=30720 Len=1460 [TCP segment of a reassembled PDU]
251	3.180502	192.168.0.15	93.93.64.123	TCP	54	50210 → 80 [ACK] Seq=644 Ack=14601 Win=65536 Len=0
309	3.489628	93.93.64.123	192.168.0.15	TCP	1514	80 → 50210 [ACK] Seq=14601 Ack=644 Win=30720 Len=1460 [TCP segment of a reassembled PDU]

> Frame 147: 66 bytes on wire (528 bits), 66 bytes captured (528 bits) on interface 0
 > Ethernet II, Src: Asustek_84:5a:55 (b0:6e:bf:84:5a:55), Dst: ArrisGro_e4:95:08 (c0:05:c2:e4:95:08)
 > Internet Protocol Version 4, Src: 192.168.0.15, Dst: 93.93.64.123
 > Transmission Control Protocol, Src Port: 50206, Dst Port: 80, Seq: 0, Len: 0

0000 c0 05 c2 e4 95 08 b0 6e bf 84 5a 55 00 00 45 00n..ZU..E..

wireshark_6954090-A00A-411B-85E1-D88E8F909E7_20181107005154_a12368.pcapng

Packets: 565 · Displayed: 35 (6.2%) · Dropped: 0 (0.0%)

Profile: Default

Código final

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

```

1  /*
2  * C++ sockets on Unix and Windows
3  * Copyright (C) 2002
4  *
5  * This program is free software; you can redistribute it and/or modify
6  * it under the terms of the GNU General Public License as published by
7  * the Free Software Foundation; either version 2 of the License, or
8  * (at your option) any later version.
9  *
10 * This program is distributed in the hope that it will be useful,
11 * but WITHOUT ANY WARRANTY; without even the implied warranty of
12 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
13 * GNU General Public license for more details.
14 *
15 * You should have received a copy of the GNU General Public License
16 * along with this program; if not, write to the Free Software
17 * Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA
18 */
19
20 #include "YASL.h" // For Socket and SocketException
21 #include "checkArgs.h"
22 #include <iostream> // For cerr and cout
23 #include <cstdlib> // For atoi()
24 #include <fstream>
25 using namespace std;
26
27 const uint32_t RCVBUFSIZE = 1024; // Size of receive buffer
28
29 int main(int argc, char *argv[]) {
30
31     checkArgs* argumentos = new checkArgs(argc, argv);
32     ofstream archivo;
33     archivo.open("fracaso.txt", ios::out);
34
35     std::string servAddress;

```

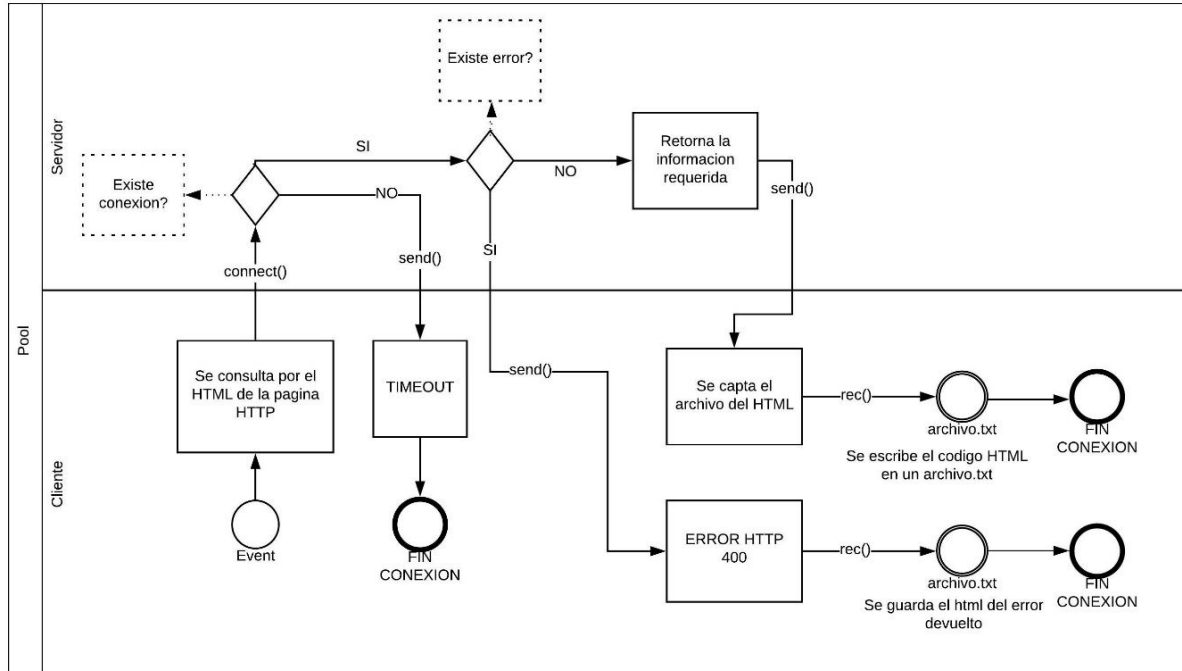
Básicamente lo que se hizo en la línea 24 y 25, fue poder implementar lo necesario para poder realizar el punto de poder guardar la página web en un archivo. Además se modificó la cantidad de bits del RCVBUFSIZE a 1024.

La otra gran modificación del código, fue poder implementar la codificación necesaria de archivos en C++, para así poder guardar todo el “echoBuffer” en archivo declarado en segmentos de códigos anteriores.

Arquitectura de software

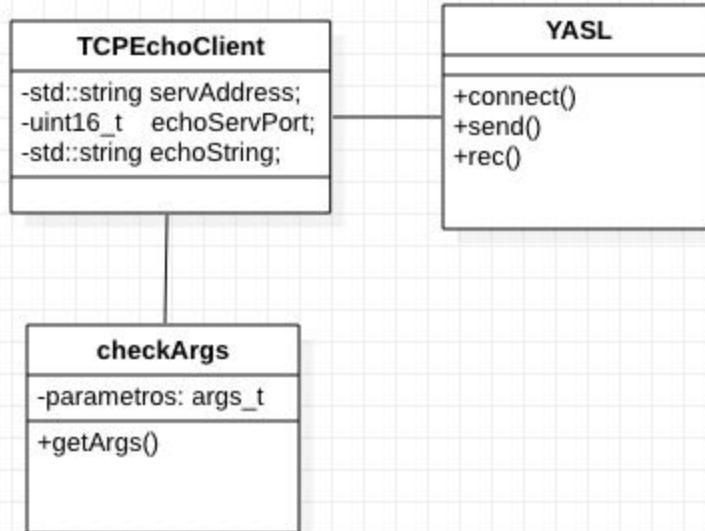
Representamos las arquitecturas del software a través del modelo 4+1 en donde tomaremos 2 “Vistas” de este modelo las cuales serán: la Vista Lógica y la Vista Proceso.

Vista Proceso



Vista lógica

Logical View
(from Untitled)



CONCLUSIÓN

El poder trabajar codificando a través de comunicaciones y no linear o secuencialmente, cambia mucho el método de trabajo, el pensamiento que uno tiene al poder codificar o poder programar de manera no online, es muy distinta a cuando el programador depende de la red para ver si su código funciona, la implementación de un cliente HTTP da a entender como es de complejo un navegador web y todo lo que tiene que hacer para que le usuario pueda ver lo que está cargando en un servidor que está a quizás cuantos kilómetros de su pantalla. En definitiva, programar dependiendo de una red, es un mundo distinto.

LINKOGRAFÍA

<http://lecturalia.com>

<http://almosthuman.cl>

<https://github.com>

<http://staruml.io>