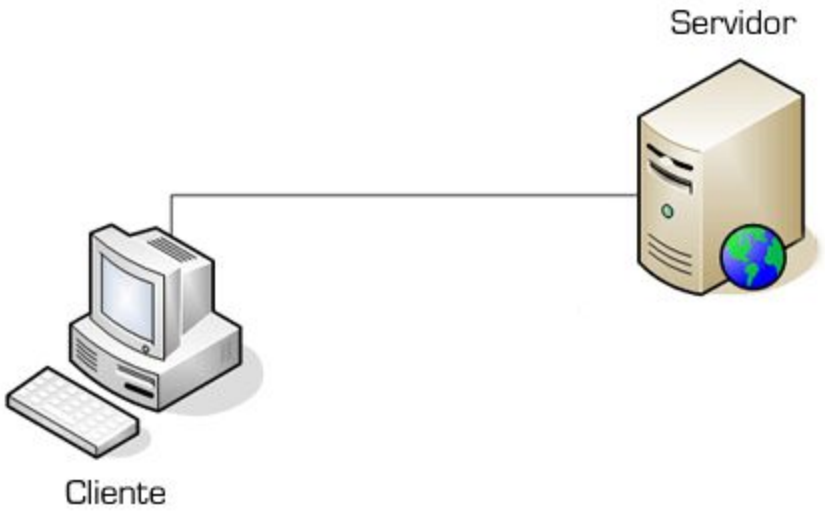
Informe Tarea 2 Redes de Computadores

Implementar servidor web TCP C++



***Integrantes*:** Manuel Carreño
Alonso Faúndez
Ariel Peña

***Profesor*:** Gabriel Astudillo

Objetivo

Implementar un software servidor web TCP sencillo en C++14, que sea capaz de servir páginas web sencillas,solo texto html, todo esto echo en sistema operativo Linux y así poder conocer y aplicar el API Networking en C++.

Desarrollo

Para implementar el servidor web iterativo simple trabajamos bajo un archivo de configuración JSON, que nos entrega la información de IP, puerto TCP, directorio de raíz donde se ubica la página web y la respuesta ante un error.

Principalmente nos enfocamos en el código del servidor, que establece la conexión con el cliente obteniendo la IP y el Puerto, y luego se encarga de enviar un mensaje respuesta.

```
int main(int argc, char *argv[]) {

    checkArgs* argumentos = new checkArgs(argc, argv);
    uint16_t echoServPort;
    echoServPort  = argumentos->getArgs().PORT;

    try {
        TCPServerSocket servSock(echoServPort);     // Server Socket object

        for (;;) {   // Run forever
            HandleTCPClient(servSock.accept());       // Wait for a client to connect
        }
    } catch (SocketException &e) {
        std::cerr << e.what() << std::endl;
        exit(EXIT_FAILURE);
    }
    // NOT REACHED

    return EXIT_SUCCESS;
}
```

```
#include "YASL.h"      // For Socket, ServerSocket, and SocketException
#include "checkArgs.h"
#include <iostream>    // For cerr and cout
#include <cstdlib>     // For atoi()
#include <fstream>

const uint32_t RCVBUFSIZE = 32;    // Size of receive buffer

// TCP client handling function
void HandleTCPClient(TCPSocket *sock) {
    std::cout << "Handling client ";
    try {
        std::cout << sock->getForeignAddress() << ":";
    } catch (SocketException e) {
        std::cerr << "Unable to get foreign address" << std::endl;
    }
    try {
        std::cout << sock->getForeignPort();
    } catch (SocketException e) {
        std::cerr << "Unable to get foreign port" << std::endl;
    }

    std::cout << std::endl;
```

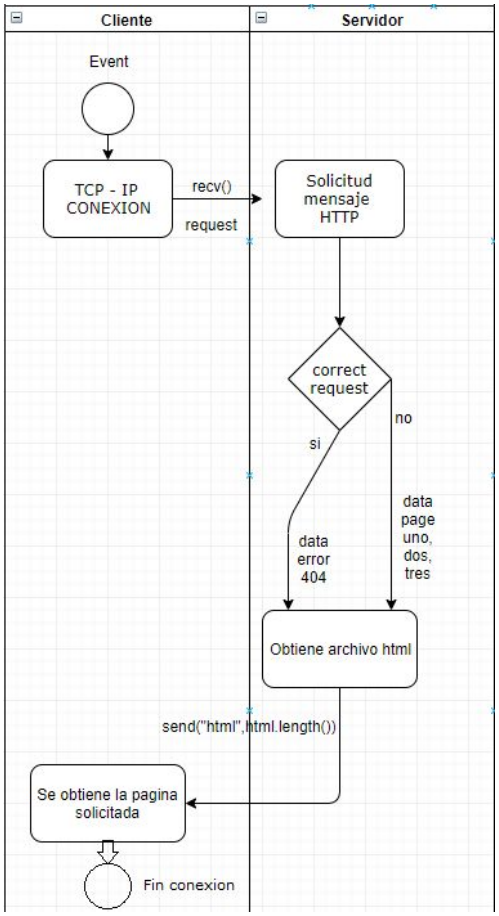
Se usa un método Try el cual es capaz tanto de hacer las peticiones de la ip y el puerto de coneccion como de manejar las excepciones en caso de que estas no resulten

```
sock->send("HTTP/1.1 200 OK\r\n",17);
sock->send("Content-Type: text/html\r\n\r\n",27);
std:: string line;
std:: string obte_html="";
std:: string obte_error="";
std:: ifstream html ("UNO.html");
std:: ifstream error ("404.html");
//lectura de html
if (html.is_open()){
    while(getline(html,line)){
        obte_html = obte_html + line + "\n";
    }
    html.close();
}else{
    //lectura de 404
    if (error.is_open()){
        while(getline(error,line)){
            obte_error = obte_error + line + "\n";
        }
        error.close();
    }
}
//envio html
sock->send(obte_html.c_str(),obte_html.length());

delete sock;
```

La modificación hecha del código fuente fue el mensaje de respuesta que se envia luego de la petición de coneccion, esta parte es la encargada de la lectura del html retornado por el servidor.Con una respuesta positiva es re-enviado el html,en caso contrario se envía un html con el encabezado de error 404

Arquitectura de Software



Vista Lógica