Universidad de Valparaíso CHILE

[1]

REDES DE COMPUTADORES: TAREA 3

Claudio Rojas, Augusto Pinochet, Sebastian Cabrera

Resumen
Se debe implementar una comunicación cliente/servidor, usando el API Networking en C++, este deberá implementar un sistema de consejos y/o tips para el desarrollo y entrega de tareas para la asignatura de Redes de Computadoras I.

Palabras clave: cliente/servidor, API, C ++,

I. INTRODUCCIÓN AL SISTEMA CREADO.

En este trabajo se implementó un tipo de comunicación cliente/servidor, mediante el lenguaje de programación C++. Este tipo de comunicación tiene el objetivo de implementar un sistema de consejos y/o tips para el desarrollo y entrega de tareas para la asignatura de Redes de Computadoras I, donde en el servidor deberá existir un archivo de texto plano en donde cada línea corresponde a un consejo de desarrollo y entrega para las tareas, cada vez que se haga un llamado, este servidor deberá enviar al cliente un consejo al azar

II. ANÁLISIS DE LA SITUACIÓN.

Inicialmente a partir de la comunicación exitosa entre el cliente y el servidor, se envía una petición de parte del cliente al servidor, para que este le entregue un consejo tomado desde el archivo consejos.txt, donde el servidor lee este archivo y toma de forma randómica solamente uno de los tantos que están en el archivo para luego responderle al cliente con el archivo requerido.

Este trabajo se implementó en el S.O. Fedora, la creación del código se realizó en editor de notas de fedora.

```c
#include<stdio.h>
#include<string.h>
#include<stdlib.h>
#include<unistd.h>
#include<arpa/inet.h>
#include<sys/socket.h>
#include<netinet/in.h>
#include<netdb.h>
#include <iostream>
#include <fstream>
using namespace std;
#include <vector>
#include<time.h>
int main(int argc, char **argv){
  if(argc<2)
  {
    printf("<host> <puerto>\n");
    return 1;
  }
  struct sockaddr_in cliente; //Se declara la conexion
  struct hostent *servidor; //se declara la informacion del host
  servidor = gethostbyname(argv[1]);
  if(servidor == NULL)
  { // Se comprueba que exista conexion
    printf("Host erróneo\n");
    return 1;
  }
  int puerto, conexion;
  char buffer[100];
  conexion = socket(AF_INET, SOCK_STREAM, 0); //Asignación del socket
  puerto=(atoi(argv[2])); //Conversion del argumento
  bzero((char *)&cliente, sizeof((char *)&cliente));
  cliente.sin_family = AF_INET;
```

Para la realización del código se utilizó los conocimientos usados en la tarea 1 y la tarea 2, por lo cual los usamos de base para empezar el desarrollo de esta tarea.

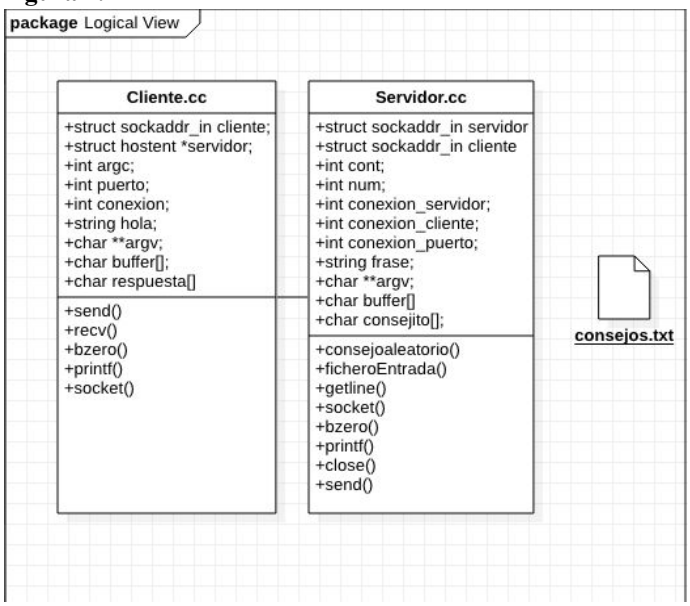
```c
  cliente.sin_port = htons(puerto);
  bcopy((char *)servidor->h_addr, (char *)&cliente.sin_addr.s_addr, sizeof(servidor->h_length));
  if(connect(conexion,(struct sockaddr *)&cliente, sizeof(cliente)) < 0)
  { //Se conecta con el host
    printf("Error conectando con el host\n");
    close(conexion);
    return 1;
  }
  printf("Conectado con %s:%d\n",inet_ntoa(cliente.sin_addr),htons(cliente.sin_port));
  char respuesta[100];
  printf("1.recibir un consejo\n");
  fgets(respuesta,100,stdin);
  if (strcmp(respuesta, "1")){
    string hola = "consejo";
    strcpy(buffer,hola.c_str());
  }
  send(conexion, buffer, 100, 0); //Se envia
  bzero(buffer, 100);
  recv(conexion, buffer, 100, 0); //Se recepciona
  printf("%s\n", buffer);
return 0;
}
```

Si el cliente logra conectar exitosamente con el servidor, este le mandara la respuesta de la petición, en forma de string y por pantalla, por lo cual mostrará el consejo al azar. El usuario define el puerto de su servidor y el ip tiene que ser 127.10.0.1.

[1]

III. VISTA LOGICA

Figura 1.



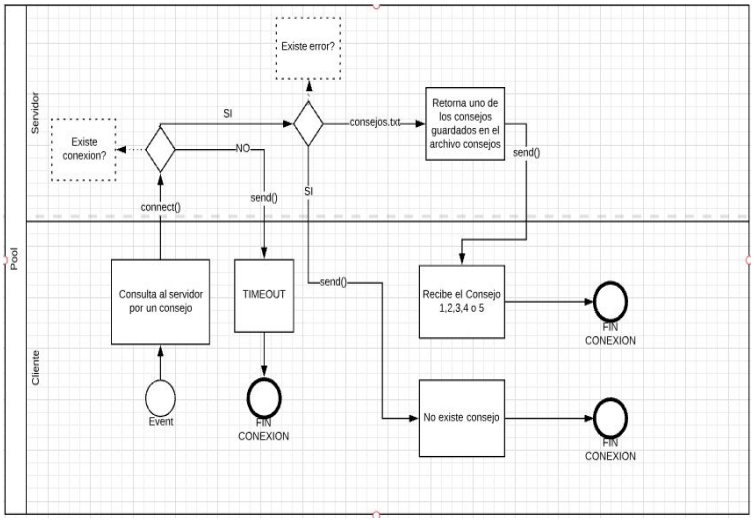
En la estructura de la figura 1, tenemos las clases cliente.cc y servidor.cc, como también el archivo consejos.txt.

En este diagrama podemos observar los atributos y operaciones importantes de cada uno, a diferencia de consejos.txt que solo sirve para obtener datos desde la clase servidor.cc

IV. VISTA DE PROCESOS

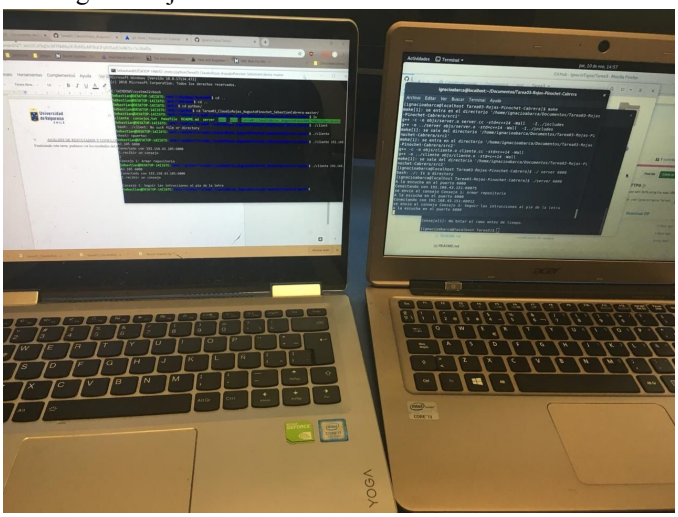
*I*nicialmente se crea el socket para definir los puerto a utilizar, el socket puede intercambiar cualquier flujo de datos que requiera el usuario a la hora de hacer las consultas al servidor, posteriormente el usuario ingresa al puerto que se desea conectar. por ejemplo el puerto 6000, y la IP a utilizar 127.10.0.1 o la ip del host en el que se quiera trabajar,en este momento es cuando el servidor está recibiendo la petición del cliente, tras recibir la petición del cliente, el servidor entrega la respuesta (el consejo a enviar aleatoriamente) y queda abierto para futuras peticiones del usuario.

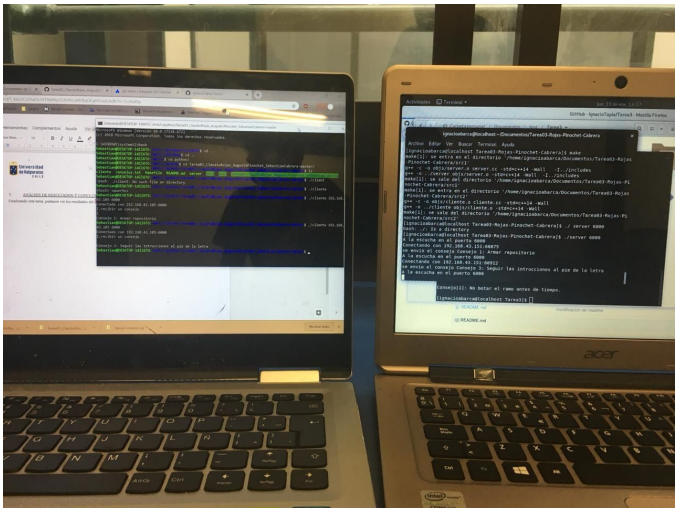
Figura 2.



V. ANÁLISIS DE RESULTADOS Y CONCLUSIONES

Finalizando esta tarea, se puede ver los resultados de las tareas anteriores en adición a esta, ya que se logra compilar el server en un computador y el cliente en otro. como se puede ver en las imágenes adjuntas.





Esto fue posible gracias al uso de algunas partes de los códigos de las tareas 1 y 2, de las cuales se obtuvo experiencia para poder trabajar con un cliente y servidor, respectivamente. Además, se ocuparon conocimientos de la asignatura de redes de computadores , las cuales se utilizó el de capa 2 para utilizar el host y los puertos, de tal manera que se logra ejecutar el código en 2 pc distintos conectados a la misma red, como se demostró anteriormente.

Ahora se puede montar un servidor simple con funcionalidades básicas, en el cual se obtendrá acceso mediante una red lan sin necesidad de utilizar alguna biblioteca externa que automatiza el trabajo.