

**Universidad Simón Bolívar**

**Departamento de Computación y Tecnología de la Información**

**Redes de Computadoras I (CI4835)**

**Trimestre Septiembre – Diciembre 2016**

**Prof. Kity Alvarez**

**Sección 1**

**Asignación 1.**

Interfaz de Aplicaciones (API) Sockets de Berkeley.

**Red de Cajeros automáticos pertenecientes al Banco Simón Bolívar**

**Integrantes:**

Cristina Betancourt 11-10104

Meggie Sánchez 11-10939

**Sartenejas, 30 de Octubre de 2016**

**ÍNDICE**

**1. Introducción** ………………………………………………………………………. 2

**2. Planteamiento del problema** …………………………………………………… 3

**3. Diseño del protocolo de comunicación** ……………………………………… 6

3.1 Tipo de socket empleado 6

3.2 Mensajes del sistema 7

3.3 Descripción del diseño del protocolo 12

3.3.1 Definición del protocolo 12

3.3.2 Modo de operación del protocolo de comunicación 12

3.4 Aspectos del planteamiento del problema modificados 14

**4. Conclusiones y Recomendaciones** …………………………………………… 16

**5. Referencias** ………………………………………………………………………... 17

**INTRODUCCIÓN**

En este trabajo expondremos todo lo relacionado a los aspectos importantes del desarrollo y escogencia de diseño de la aplicación final, donde usaremos el paradigma de programación Cliente – Servidor, para así de esta manera, lograr el entendimiento o comprensión de las aplicaciones y servicios en redes.

Esta aplicación final, es una simulación de una red de cajeros automáticos ubicados en la Universidad Simón Bolívar, ubicados en la planta baja de los edificios: Básico I, Básico II y ENE, donde sabemos que estos cajeros permitirán realizar transacciones como depósitos y retiros, además, se incluye un Computador Central (CC) que es el que va a controlar el sistema.

Así, con el paradigma Cliente – Servidor, los cajeros automáticos simulan ser los clientes y el computador central simula ser el servidor. Con ello, se desea comprender el uso y la programación de la Interfaz de Aplicaciones (API) Sockets de Berkeley, y de igual forma, aplicar los conceptos dados en clases para desarrollar el diseño e implementación de un protocolo de comunicación básico.

**PLANTEAMIENTO DEL PROBLEMA**

La Universidad Simón Bolívar, dispone de una red de cajeros automáticos pertenecientes al Banco Simón Bolívar, éstos, se encuentran ubicados en la planta baja de los edificios: Básico I, Básico II y ENE, permiten realizar transacciones diarias de depósito y retiro a los clientes. (Son cajeros tipo full).

Se quiere diseñar un sistema informático basado en el paradigma Cliente/Servidor que permita automatizar y controlar dicha red. Para ello se le ha contactado a usted como estudiante del curso de Redes de Computadores I, a fin que proponga e implemente una aplicación sencilla que incluya un Computador Central (CC) que va a controlar el sistema.

**Funcionamiento del Sistema**

* El prototipo debe simular el funcionamiento del sistema las 24 horas del día, los 7 días a la semana (24/7).
* El “*Total Disponible*” para movimientos por cajero es de Bs. 80.000.
* Cuando se disponga de un “*Total Disponible*” con menos de Bs. 5.000, deberá enviar un mensaje a la central comunicando la situación a fin de ser recargado el monto por dispositivo.
* Si el cajero cuenta con un “*Total Disponible*” igual o inferior a Bs. 5.000, se informa al usuario de dicho evento. En la pantalla deberá mostrarse un mensaje de “*Dinero No Disponible*” dependiendo de los retiros que desee realizar.
* Al cierre del día el Computador Central debe tener un registro de todas las operaciones realizadas indicando fecha, hora, código del usuario, el evento (Depósito/Retiro) y Total Disponible.
* Cuando el CC recibe un mensaje de “*Retiro*” se decrementa el contador de *Total Disponible* y responde al cajero adecuado con un mensaje indicando la hora y fecha de retiro, un código (número entero) que identifica al usuario.
* Con la fecha, hora, operación y código, el cajero automático imprime el ticket del usuario.
* El usuario podrá realizar máximo tres (3) retiros por un monto máximo de Bs. 3.000 cada uno.
* Cuando un usuario desea realizar un “*Depósito*”, introduce el código de usuario y monto del depósito.
* Cada vez que un usuario deposita en su cuenta, el CC debe incrementar el “*Total Disponible”* en cajero e imprimir ticket con la información (fecha, hora, operación y código).
* Considere que puede haber servicios simultáneos.

.

**DISEÑO DEL PROTOCOLO DE COMUNICACIÓN**

**3.1 TIPO DE SOCKET EMPLEADO**

**Para saber el tipo de socket empleado, primero dejaremos una pequeña definición de socket: “Un socket es un mecanismo que permite la conexión entre distintos procesos**, habitualmente se utilizan para establecer comunicaciones entre distintas máquinas que estén conectadas a través de la red”. Este socket, es una interfaz del modelo de protocolo TCP/IP.

La llamada al socket posee tres parámetros, el primero es el dominio, donde usamos AF\_INET, ya que se refiere a que se permite el intercambio de paquetes en Internet y en distintas computadoras. El segundo es el tipo, donde usamos SOCK\_STREAM, ya que este tipo de conexión proporciona secuencias confiables, y de dos vías. Además, un mecanismo de transmisión de datos fuera de banda puede estar soportado. Por último, el tercer parámetro es el protocolo, el cual es TCP.

La razón por la que decidimos utilizar el protocolo TCP es porque primeramente, es orientado a conexión, y como la aplicación final es la simulación de una red de cajeros automáticos, se necesita de una conexión entre el cajero y el computador central, y garantizar no sólo que la conexión se realizó, sino que al hacer un depósito o retiro, el cliente pueda recibir su factura o confirmación de que dicha operación si se logró, detalle que cumple este protocolo ya que cuando los paquetes llegan al receptor este emite un mensaje de recepción (ACK). Además, garantiza la correcta transmisión de los paquetes y mantiene el orden de los mismos.

**3.2 MENSAJES DEL SISTEMA**

En lo referente a los mensajes del sistema, pondremos a continuación todos los mensajes del sistema, indicando: el formato del mismo, su tamaño en bytes, quién genera el mensaje y quién lo recibe y procesa.

**Punto 1:** Una vez ya establecida la conexión entre cliente y servidor, el servidor le manda un mensaje al cliente para darle la bienvenida.

**Mensaje 1:** “ Mensaje del Servidor: Bienvenido a mi servidor ”

Formato: ASCII

Tamaño en Bytes: 25 bytes

Mensaje generado por: Computador Central (Servidor)

Mensaje recibido por: Cajero (Cliente)

Mensaje procesado por: Socket TCP

**Punto 2:** Una vez ya establecida la conexión con 3 clientes y el servidor, el servidor manda un mensaje al 4to cliente que intenta conectarse.

**Mensaje 2:** “ Mensaje del Servidor: Todos los cajeros estan en uso. Intente mas tarde ”

Formato: ASCII

Tamaño en Bytes: 51 bytes

Mensaje generado por: Computador Central (Servidor)

Mensaje recibido por: Cajero (Cliente)

Mensaje procesado por: Socket TCP

**Punto 3:** Una vez que el mismo cliente o diferentes clientes, quieren retirar más de 3 veces, se imprime un mensaje indicando que se excedió el número de veces de retiro.

**Mensaje 3:** “ Ya se ha excedido de la cantidad de veces de retiro “

Formato: ASCII

Tamaño en Bytes: 10 bytes

Mensaje generado por: Computador Central (Servidor)

Mensaje recibido por: Cajero (Cliente)

Mensaje procesado por: Socket TCP

**Punto 4:** Una vez ya establecida la conexión del cliente con el servidor, el cliente imprime un mensaje esperando por el monto que el usuario desea retirar.

**Mensaje 4:** “ Introduzca el monto a retirar: ”

Formato: ASCII

Tamaño en Bytes: 512 bytes (buffer recibido con tamaño alto por si se desea depositar una cantidad alta de dinero) (aunque teóricamente no se puede)

Mensaje generado por: Cajero (Cliente)

Mensaje recibido por: Computador Central (Servidor)

Mensaje procesado por: Socket TCP

**Punto 5:** Una vez ya establecida la conexión del cliente con el servidor, el cliente imprime un mensaje esperando por el monto que el usuario desea depositar.

**Mensaje 5:** “ Introduzca el monto a depositar: ”

Formato: ASCII

Tamaño en Bytes: 512 bytes (buffer recibido con tamaño alto por si se desea depositar una cantidad alta de dinero)

Mensaje generado por: Cajero (Cliente)

Mensaje recibido por: Computador Central (Servidor)

Mensaje procesado por: Socket TCP

**Punto 6:** Cuando el cliente desea retirar una cantidad mayor de 3000 Bs, el cajero imprime un mensaje indicando que esto no se puede.

**Mensaje 6:** “ El monto excede la cantidad permitida de Bs. 3000 “

Formato: ASCII

Tamaño en Bytes: 512 bytes (buffer recibido con tamaño alto por si se desea depositar una cantidad alta de dinero) (aunque teóricamente no se puede)

Mensaje generado por: Cajero (Cliente)

Mensaje recibido por: Computador Central (Servidor)

Mensaje procesado por: Socket TCP

**Punto 7:** El cajero imprime un mensaje de retiro realizado cuando el retiro no pasa de Bs 3000 y el total disponible es mayor que Bs 5000.

**Mensaje 7:** “ El retiro se ha realizado satisfactoriamente. “

Formato: ASCII

Tamaño en Bytes: 512 bytes (buffer recibido con tamaño alto por si se desea retirar una cantidad alta de dinero) (aunque teóricamente no se puede)

Mensaje generado por: Cajero (Cliente)

Mensaje recibido por: Computador Central (Servidor)

Mensaje procesado por: Socket TCP

**Punto 8:** El cajero imprime un mensaje de deposito cuando el deposito se ha realizado satisfactoriamente.

**Mensaje 8:** “ El deposito se ha realizado satisfactoriamente. ”

Formato: ASCII

Tamaño en Bytes: 512 bytes (buffer recibido con tamaño alto por si se desea depositar una cantidad alta de dinero)

Mensaje generado por: Cajero (Cliente)

Mensaje recibido por: Computador Central (Servidor)

Mensaje procesado por: Socket TCP

**Punto 9:** Si el cajero cuenta con un “*Total Disponible*” igual o inferior a 5.000 bs, se informa al usuario de dicho evento. En la pantalla deberá mostrarse un mensaje de “*Dinero No Disponible*” dependiendo de los retiros que desee realizar.

**Mensaje 9:** “ Dinero No Disponible ”

Formato: ASCII

Tamaño en Bytes: 512 bytes

Mensaje generado por: Computador Central (Servidor)

Mensaje recibido por: Cajero (Cliente)

Mensaje procesado por: Socket TCP

**Punto 10:** Al cierre del día el Computador Central debe tener un registro de todas las operaciones realizadas indicando fecha, hora, código del usuario, el evento (Depósito/Retiro) y Total Disponible.

**Mensaje 10:**

“ Fecha: dd/mm/aaaa

Hora: hh:mm

Codigo de usuario: cccc

Monto: x

Total Disponible: x ”

Formato: ASCII

Tamaño en Bytes: 21 bytes mínimo, en adelante (dependiendo del monto introducido y del total disponible)

Mensaje generado por: Computador Central (Servidor)

Mensaje recibido por: Computador Central (Servidor)

Mensaje procesado por: Socket TCP

**Punto 11:** Cuando el CC recibe un mensaje de “*Retiro*” decrementa el contador de *Total Disponible* y responde al cajero adecuado con un mensaje indicando la hora y fecha de retiro, un código (número entero) que identifica al usuario.

**Mensaje 11:**

“ Fecha: dd/mm/aaaa

Hora: hh:mm

Codigo de usuario: cccc ”

Formato: ASCII

Tamaño en Bytes: 19 bytes

Mensaje generado por: Computador Central (Servidor)

Mensaje recibido por: Cajero (Cliente)

Mensaje procesado por: Socket TCP

**Punto 12:** Cada vez que un usuario deposita en su cuenta, el CC debe incrementar el “*Total Disponible”* en cajero e imprimir ticket con la información (fecha, hora y código)

**Mensaje 12:**

“ Fecha: dd/mm/aaaa

Hora: hh:mm

Codigo de usuario: cccc ”

Formato: ASCII

Tamaño en Bytes: 19 bytes

Mensaje generado por: Computador Central (Servidor)

Mensaje recibido por: Cajero (Cliente)

Mensaje procesado por: Socket TCP

**3.3 DESCRIPCIÓN DEL DISEÑO DEL PROTOCOLO**

**3.3.1 Definición del protocolo de comunicación**

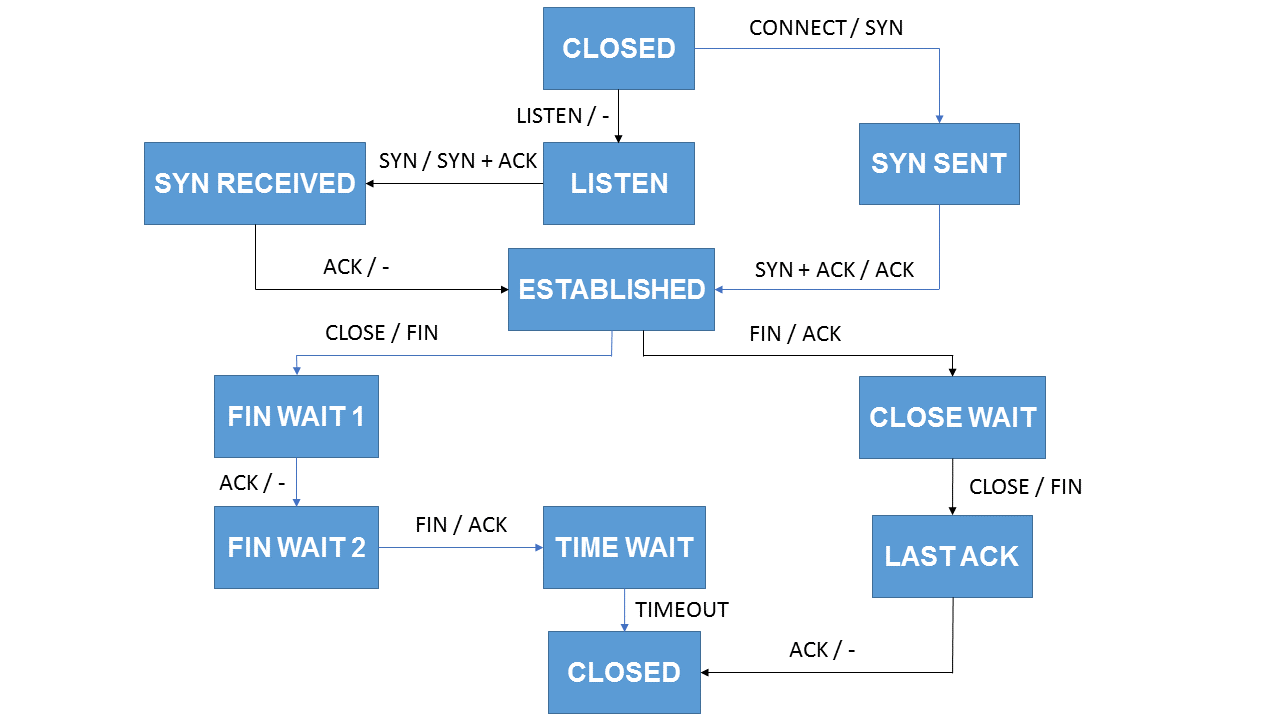
Como protocolo de comunicación utilizado, escogimos el protocolo TCP (Protocolo de Control de Transmisión). En la pila de protocolos TCP/IP, TCP es la capa intermedia entre el protocolo de red (IP) y la aplicación. Muchas veces las aplicaciones necesitan que la comunicación a través de la red sea confiable. El protocolo TCP asegura que los datos que emite el cliente sean recibidos por el servidor sin errores y en el mismo orden que fueron emitidos, a pesar de trabajar con los servicios de la capa IP, la cual no es confiable. Es orientado a la conexión, ya que el cliente y el servidor deben anunciarse y aceptar la conexión antes de comenzar a transmitir los datos a ese usuario que debe recibirlos.

**3.3.2 Modo de operación del protocolo de comunicación**

Para mayor entendimiento, ilustraremos un esquema donde se podrá observar la conexión TCP con un diagrama de estados explicado previamente en clases, luego, se explicará con un poco más de detalle.

|  |  |
| --- | --- |
| **Estado** | **Descripción** |
| CLOSED | No hay conexión activa ni pendiente |
| LISTEN | El servidor espera una llamada |
| SYN RCVD | Llegó solicitud de conexión; espera ACK |
| SYN SENT | La aplicación comenzó a abrir una conexión |
| ESTABLISHED | Estado normal de transferencia de datos |
| FIN WAIT 1 | La aplicación dijo que ya terminó |
| FIN WAIT 2 | El otro lado acordó liberar |
| TIMED WAIT | Espera que todos los paquetes mueran |
| CLOSE WAIT | El otro lado inició una liberación |
| LAST ACK | Espera que todos los paquetes mueran |

**Estados usados en la máquina de estados finitos de administración de conexiones TCP.**



**Máquina de estados finitos de administración de conexiones TCP. La línea continua gruesa azul es la trayectoria normal de un cliente. La línea punteada gruesa negra es la trayectoria normal de un servidor. Cada transición está indicada por el evento que la ocasiona y la acción resultante, separada por una diagonal.**

Para entender el diagrama, primero analizaremos la trayectoria de un cliente (línea clara) y luego la de un servidor (línea oscura).

Al emitir una solicitud CONNECT una aplicación de la máquina del cliente, la entidad TCP local crea un registro de conexión, lo que indica que está en el estado SYN SENT y envía un segmento SYN. Al legar el SYN+ACK, el TCP envía el ACK final de acuerdo de tres vías y se conmuta al estado ESTABLISHED. A partir de ahora pueden enviarse y recibirse datos.

Al terminar una aplicación, esta ejecuta una primitiva CLOSE que causa que la entidad TCP local envíe un segmento FIN y espere el ACK correspondiente. Cuando llega este ACK se hace una transición del estado FIN WAIT 2 y ya está cerrado en un sentido. Cuando también cierra el otro lado, llega un FIN, para el cual se envía una confirmación de recepción y se cierran los dos lados, pero el TCP espera un tiempo igual al tiempo de vida máximo del paquete para garantizar que todos los paquetes de la conexión han sido eliminados. Cuando termina el temporizador, el TCP borra el registro de la conexión.

En cuanto a la conexión del servidor, este hace un LISTEN y se detiene a esperar la aparición de alguien. Al llegar un SYN, se envía una confirmación y el servidor pasa al estado SYN RCVD. Cuando llega la confirmación de recepción del SYN servidor, se completa el acuerdo de tres vías y se pasa al estado ESTABLISHED, que permite la transferencia de datos. Cuando el cliente hace un CLOSE, causa la llegada de un FIN al servidor. Cuando este también hace un CLOSE, se envía un FIN al cliente. Al llegar la confirmación de recepción, el servidor libera la conexión y elimina el registro de conexión.

**3.4 Aspectos del planteamiento del problema modificados**

**Primera modificación**

En cuanto a la segunda condición del funcionamiento del sistema que se describe en el capítulo 2 de este informe, se indica que el “Total Disponible” por cajero es de Bs. 80.000. De acuerdo con lo discutido en clase, al no ser necesario una cuenta por cajero, este total le corresponde al servidor, por lo que es una variable global que se incrementa o decrementa por depósitos o retiros respectivamente sin importar de que cajero provenga la operación.

**Segunda modificación**

Con respecto a la tercera y cuarta condición del funcionamiento del sistema, que hablan sobre los mensajes que se envían al servidor y al cliente cuando el “Total Disponible” es igual o menor a Bs. 5.000, dado a que ahora sólo se cuenta con el mismo monto para los tres cajeros y que este monto lo posee el servidor, los clientes no tienen que notificar nada al servidor. Simplemente el servidor envía un mensaje cuando el Total Disponible sea menor a Bs. 5.000 indicando al cliente que no puede retirar.

**Tercera modificación**

Referente a las bitácoras de depósito y retiro, se crea un archivo de texto para cada uno, donde por cada operación se especifica la fecha, hora, código de usuario, el “Total Disponible” y, además, consideramos necesario agregar el monto retirado o depositado, para que al momento de analizar el historial, se entendieran perfectamente las cuentas. En cuanto a la operación, nos pareció irrelevante colocarla dado a que se especifica un archivo para los depósitos y un archivo para los retiros.

**Cuarta modificación**

Por último, en la novena condición del funcionamiento del sistema, se dice que para realizar un depósito se debe introducir el monto y el código de usuario, pero, dado a que cuando se ingresa la información en la línea de comando para conectar con el servidor ya se registra el código de usuario, es innecesario volverlo a introducir.

**CONCLUSIONES Y RECOMENDACIONES**

Con el presente proyecto se obtuvo el aprendizaje del uso e implementación de la API de Sockets de Berkeley, a través del simulador de red de los cajeros automáticos. Este objetivo se logró alcanzar el entendimiento del paradigma Cliente – Servidor y específicamente con el lenguaje de programación C.

Como posibles recomendaciones, se podría aumentar el número de servicios ofrecidos por el servidor, como por ejemplo, consultar el total disponible, para que así el usuario se pueda ahorrar retiros fallidos por monto insuficiente.

Otra recomendación podría ser que al imprimir el ticket en el cajero, se muestre el total disponible, para que así se pueda llevar la cuenta de cuánto es el saldo disponible luego de realizada la operación de depósito o retiro.

**REFERENCIAS BIBLIOGRÁFICAS**

1. Sintaxis de las primitivas de la API de *sockets*. Redes de Computadoras 4 Edición Andrew S. Tanenbaum
2. https://moodle.asignaturas.usb.ve/pluginfile.php/38266/mod\_folder/content/0/Sockets\_RedesI\_Sept\_Dic2016.pdf
3. http://ldc.usb.ve/~rgonzalez/ci4835/taller/ClasedeSocketsenlenguajeC\_SoloSockets.pdf
4. http://ldc.usb.ve/~poc/Redes/PDF/ModeloCS.pdf
5. http://ldc.usb.ve/~figueira/cursos/redes/ejemplos/