Universidad Simón Bolívar

Departamento de Computación y Tecnología de la Información

Redes de Computadores

# INFORME PROYECTO SOCKETS

Nombre y Carnet:

Arleyn Goncalves 10-10290

Francisco Sucre 10-10717

Sartenejas, 30 de Octubre del 2016

# ÍNDICE

# INTRODUCCIÓN

Este informe tiene como propósito plantear la solución y el diseño de un sistema informático basado en el paradigma Cliente/Servidor para automatizar y controlar una red de cajeros automáticos pertenecientes al Banco Simón Bolívar, estos se encuentran ubicados en la planta baja de los edificios: Básico I, Básico II y EME de la Universidad Simón Bolívar.

Para la implementación del sistema se considera el uso de sockets para la comunicación entre el cliente y el servidor, es decir, los cajeros automáticos y el sistema central que trabaja la red, indicando el flujo de entrada y salida, se establecerá el protocolo TCP (*Transmission Control Protocol)* como capa de transporte.

## CONTENIDO

### Planteamiento del Problema

Se busca diseñar un sistema informático que permita automatizar y controlar la comunicación entre un cajero automático y un sistema central que trabaja en la red, se disponen de tres cajeros automático que tienen un puerto de comunicaciones multiprotocolo que le permite conectarse por módem (telefónico, de radio o celular), o vía satélite, una impresora de entrega de recibos, un sistema con un lector de bandas magnéticas, un dispensador de billetes y un sensor que reconoce las diferentes denominaciones o documentos que maneja la máquina.

Los tres cajeros tienen funciones completas, es decir, que se permiten realizar depósitos, retiros y otras transacciones como consignaciones, pagos y transferencias, entre otros. Además el cajero cuenta con un sistema de auditoría interna en el disco duro que permite verificar las transacciones sin que el cajero salga de servicio.

El sistema se modelará como un paradigma Cliente/Servidor el cual controlará por completo la red de cajeros automáticos pertenecientes al Banco Simón Bolívar.

### Objetivo General

Entender, de manera general, el funcionamiento simple de aplicaciones y servicios en redes mediante la implementación del paradigma de programación Cliente/Servidor.

### Objetivos Específicos

1. Desarrollar el diseño e implementación de un protocolo de comunicación básico.
2. Comprender el uso y la programación de la Interfaz de Aplicaciones (API) Sockets de Berkeley.

### Diseño del Protocolo de Comunicación

La red de cajeros automáticos esta constituido principalmente por un Computador Central (CC) que va a controlar el sistema y 3 cajeros automáticos, haciendo una analogía con el paradigma Cliente/Servidor, donde el Computador Central es el servidor y lo 3 cajeros son los clientes.

Para efectuar la comunicación entre el (CC) y los cajeros automáticos se implementó en la capa de transporte el protocolo TCP (*Transmission Control Protocol* o Protocolo de Control de Transporte), este protocolo 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, también asegura que los datos serán entregados a su destino sin errores y en el mismo orden en que se transmitieron.

### Socket Implementado

Para la comunicación entre los clientes y el servidor se decidió implementar para la capa de transporte los sockets de flujo que utilizan el protocolo TCP, ya que nos aseguran que todos los datos trasmitidos lleguen sin errores ni omisiones y que todos los datos llegarán a su destino en el mismo orden en que se han transmitido. Para su implementación se debe utilizar los atributos AF_INET para protocolos de Internet IPv4 y SOCK_STREAM que nos permiten comunicaciones fiables en modo conectado y tiene definidas las siguientes propiedades: ningún dato transmitido se pierde, los datos llegan en el orden en el que han sido emitidos, no se duplican los datos y se establece una conexión entre dos puntos antes del principio de la comunicación, estas propiedades son vitales para la eficiencia y confiabilidad del sistema.

### Mensajes del Sistema

Para el buen funcionamiento del sistema, se necesita que exista comunicación entre el cliente y el servidor, que consistirá en un intercambio de mensajes entre ellos que dependerá de las acciones que quiera realizar el cliente (depósito o retiro de dinero) y de la respuesta del servidor ante la petición del cliente.

Estos mensajes se envían con el formato <Operacion>-<Id De Usuario>-<Monto>, en un string

de tamaño estático de 700 caracteres, se escogió esta implementación debido a las sencillez de la solución y que la optimización de memoria no era parte de los objetivos del proyecto. Sabiendo este formato el servidor toma los datos de este mensaje uno por uno y luego efectúa la operación requerida.

**Provenientes del Cliente**

- Mensaje para realizar un deposito: en este mensaje el cliente ingresa el tipo de operación a realizar en este caso un depósito, el monto y el identificador del usuario.

**Flujo del mensaje:**

Genera el mensaje: Cliente

Recibe el mensaje: Servidor

Procesa el mensaje: Servidor

**Formato del mensaje:**

Operación: d

Monto: cantidad de dinero a depositar.

UserId: identificador del usuario.

- Mensaje para realizar un retiro: en este mensaje el cliente ingresa el tipo de operación a realizar en este caso un retiro, el monto y el identificador del usuario.

**Flujo del mensaje:**

Genera el mensaje: Cliente

Recibe el mensaje: Servidor

Procesa el mensaje: Servidor

**Formato del mensaje:**

Operación: r

Monto: cantidad de dinero a retirar

UserId: identificador del usuario

**Provenientes del Servidor**

- Mensaje cuando se hace efectivo el depósito: luego de que el cliente envía la petición para realizar el depósito, el servidor verifica que el usuario no este realizando un retiro o depósito en otro cajero. Si la transacción es exitosa el servidor responde con un Ticket donde se encuentran los datos de la operación.

**Flujo del mensaje:**

Genera el mensaje: Servidor

Reciba el mensaje: Cliente

Procesa el mensaje: Cliente

**Formato del Mensaje:**

Operación: Depósito

Date: fecha en la que se realizó el depósito

Time: hora en la que se realizó el depósito

userCode: identificador del usuario que realiza el depósito

- Mensaje cuando se hace efectivo el retiro: luego de que el cliente envía la petición para realizar el retiro, el servidor verifica que el usuario no este realizando un retiro o depósito en otro cajero y que el cajero tenga suficiente dinero para realizar la operación. Si la transacción es exitosa el servidor responde con un Ticket donde se encuentran los datos de la operación.

**Flujo del mensaje:**

Genera el mensaje: Servidor

Reciba el mensaje: Cliente

Procesa el mensaje: Cliente

**Formato del Mensaje:**

Operación: Retiro

Date: fecha en la que se realizó el retiro

Time: hora en la que se realizó el retiro

userCode: identificador del usuario que realiza el retiro

**Mensajes de información provenientes del Servidor**

- Cuando el monto de retiro es mayor que el monto que tiene el servidor: luego de que el usuario envía la petición de retiro, el servidor verifica la información y que el monto solicitado por el usuario este disponible en el cajero automático, en caso de no ser así el servidor le emite un mensaje al cliente indicando que el dinero no está disponible.

**Flujo del Mensaje:**

Genera el mensaje: Servidor

Reciba el mensaje: Cliente

Procesa el mensaje: Cliente

- Cuando el usuario ya ha realizado tres retiros: luego de que el usuario solicita hacer un retiro, el servidor verifica cuantos retiros ha realizado el usuario, si ya tiene tres retiros exitosos, el servidor emite un mensaje indicando que ya no se pueden hacer mas operaciones por el día de hoy.

**Flujo del Mensaje:**

Genera el mensaje: Servidor

Reciba el mensaje: Cliente

Procesa el mensaje: Cliente

**Comunicación en el Protocolo**

A continuación se presentará la descripción del diseño completo del protocolo de comunicación que realiza el cliente con el servidor, planteando los dos posibles escenarios cuando el usuario quiere realizar un depósito y cuando quiere realizar un retiro:

**- Se realiza un depósito:**

1. El cliente emite una petición al servidor para realizar un depósito, envía un mensaje indicando que operación desea realizar en este caso es un depósito, el monto a depositar y el código identificador del usuario.

2. Una vez enviado el mensaje, el cliente queda esperando respuesta del servidor, si se logra realizar la operación el servidor emite un ticket como respuesta sino envía un mensaje indicando que la operación no pudo ser realizada.

3. Si la respuesta es exitosa el cliente recibe un ticket con los datos del depósito (fecha, hora, tipo de operación y código de usuario), en caso que no se pueda realizar la operación porque no se logro conectar con el servidor, este envía un mensaje al cliente y finaliza; si el cliente quiere intentarlo de nuevo debe realizar una nueva petición desde el paso 1.

4. Cuando el cliente se conecta con el servidor entra en modo de espera, mientras tanto el servidor realiza las verificaciones y operaciones correspondientes para poder realizar el depósito, al finalizar el servidor puede responder con un ticket o con un mensaje de información.

5. Luego de que el servidor envía respuesta a la petición del cliente, queda en modo de espera por una nueva petición por parte de algún cliente.

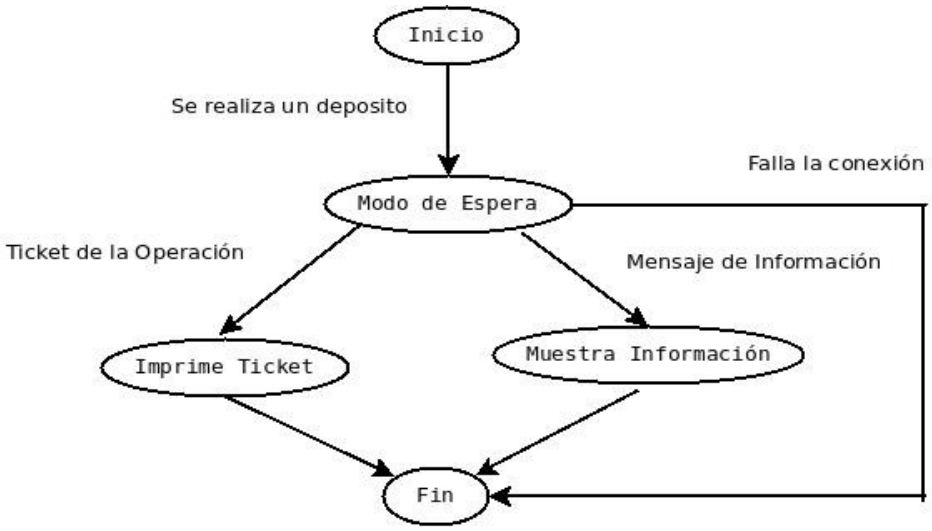


Diagrama de Estado (Cliente realiza un deposito)

**- Se realiza un retiro:**

1. El cliente emite una petición al servidor para realizar un retiro, envía un mensaje indicando

que operación desea realizar en este caso es un retiro, el monto a retirar  y el código identificador del usuario.

2.  Una vez enviado el mensaje, el cliente queda esperando respuesta del servidor, si se logra realizar la operación el servidor emite un ticket como respuesta sino envía un mensaje indicando que la operación no pudo ser realizada.

3. Si la respuesta es exitosa el cliente recibe un ticket con los datos del retiro (fecha, hora, tipo de operación y código de usuario), en caso que no se pueda realizar la operación porque no se logro conectar con el servidor, este envía un mensaje al cliente y finaliza; si el cliente quiere intentarlo de nuevo debe realizar una nueva petición desde el paso 1.

4.  Cuando el cliente se conecta con el servidor entra en modo de espera, mientras tanto el servidor realiza las verificaciones y operaciones correspondientes para poder realizar el retiro, al finalizar el servidor puede responder con un ticket o con un mensaje de información.

5. Luego de que el servidor envía respuesta a la petición del cliente, queda en modo de espera por una nueva petición por parte de algún cliente.

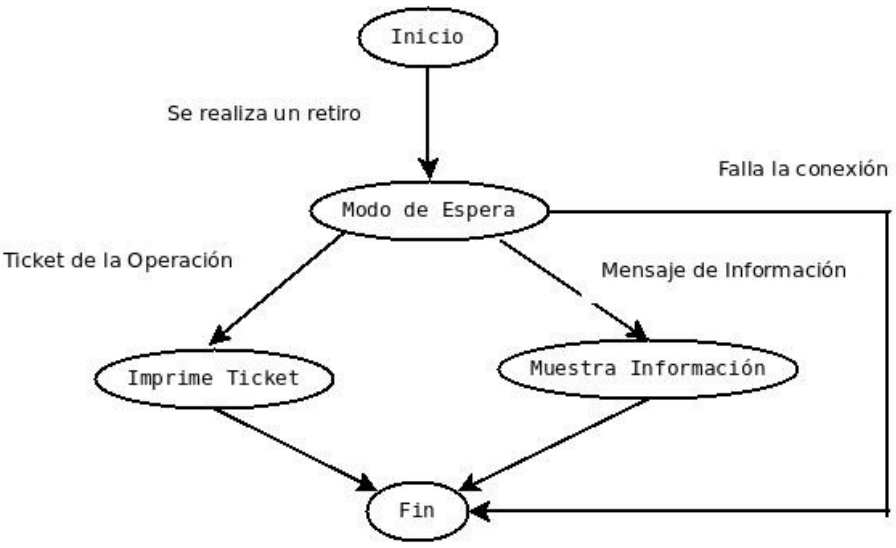


Diagrama de Estado (Cliente realiza un retiro)

**- Estados del Servidor**

1. Se inicia el servidor y queda en modo de espera hasta que llegue una petición de algún cliente.

2. Cuando llega una petición el servidor, se realizan la operaciones y verificaciones correspondientes, si el resultado es exitoso se emite un ticket y si falla se envía un mensaje de información.

3. En dado caso que llegue una petición mientras se está procesando otra, está se encola hasta que pueda ser atendida. Las peticiones se van atendiendo por orden de llegada.

4. Una vez que se finalicen todas las peticiones, el servidor entra en modo de espera.

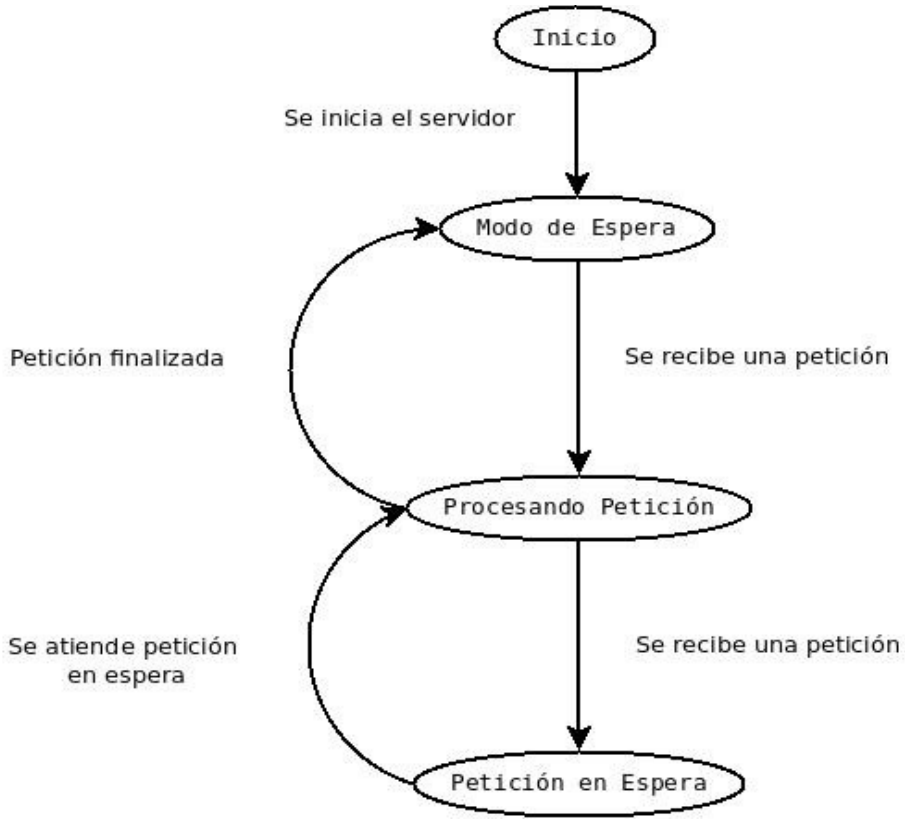


Diagrama de Estado del Servidor

**Consideraciones adicionales**

- Al implementar el sistema se tomo en cuenta que podían llegar al servidor varias solicitudes al mismo tiempo, para poder manejar la concurrencia se implementaron hilos, cada cajero tiene su propio hilo.

- El sistema identifica y registra cajeros en la red dependiendo de la dirección IP desde que se conecta el cliente, es decir, inicialmente se tienen registrados tres cajeros, en dado caso que el servidor tenga una petición de un cajero que no este registrado, este cajero pasa a ser registrado, el sistema esta implementado para un numero infinitos de cajeros.

- Inicialmente cada cajero tiene 80.000 BsF.

- Si el usuario realiza una petición y no llega la petición al servidor, se envía un mensaje de que no se pudo conectar y si se desea repetir la operación se debe realizar el proceso desde el inicio.

- El sistema verifica que se ingresen todos los parámetros correspondientes, además pueden ser invocados en un orden diferente al que se indicó inicialmente en la sintaxis.

# CONCLUSIÓN

La implementación del sistema para una red de cajeros nos ha permitido aprender y profundizar sobre el paradigma Cliente/Servidor, como funciona y como establecer la comunicación mediante los socktes de Berkeley.

Al iniciar la implementación del sistema se tuvo que elegir que protocolo usar para la capa de transporte, se decidió usar el protocolo TCP ampliando nuestros conocimientos sobre los protocolos de comunicación, como funcionan, y cuales son las ventajas y desventajas que tienen unos protocolos sobre otros, mostrándonos su importancia para el desarrollo de las redes de comunicación.

Finalmente se logró desarrollar un sistema bastante sólido y que cumple con todos los requerimientos solicitados, logrando los objetivos del proyecto y de este modulo del curso.

## REFERENCIAS BIBLIOGRÁFICAS

- Computer Networks (Quinta Edición) por Andrew S. Tanenbaum y David J. Wetheral

- http://www.linuxhowtos.org/C_C++/socket.htm

- http://es.tldp.org/Tutoriales/PROG-SOCKETS/prog-sockets.html