Universidad Simón Bolívar

Departamento de Computación y Tecnología de la Información

Redes de Computadoras I - CI4835

ESTACIONAMIENTO C.C. MORIAH

Integrantes:

Alejandra Cordero / Carnet: 12-10645

Ricardo Mena / Carnet: 12-10872

Abril - Julio 2016

Tabla de contenidos

1 . Introducción

En el presente trabajo se presenta el diseño un protocolo de comunicación básico que hace uso de la Interfaz de Aplicación Sockets UDP.

Este protocolo será utilizado en un sistema informático basado en el paradigma cliente/servidor que permitirá controlar el acceso y la salida de los vehículos que hacen uso de estacionamiento del Centro Comercial Moriah.

2. Tipo de sockets utilizados.

El tipo de socket utilizados para implementar el protocolo de comunicación que será utilizado en el estacionamiento Moriah fue el socket UDP.

Se utilizó este tipo de socket y no el TCP por varias razones:

TCP se utiliza en la red porque es un protocolo confiable que evita en gran medida la pérdida de información, sin embargo, el tamaño de los paquetes que se envían a través del protocolo, su sencillez y la falta de secuenciación del mismo hace que utilizar TCP sea innecesario, esto sumado al hecho de que el sistema funcionará solamente en una red local lo cual reduciría considerablemente la probabilidad de alguna pérdida de información.

Utilizar TCP en este caso seria no seria lo ideal ya es una pérdida de recursos crear una conexión TCP para enviar un paquete de unos cuantos bytes . Además, el establecimiento de esta conexión hace que el envío de información posea un cierto nivel de retardo. No obstante UDP, realiza un envío más rápido.

Para este protocolo la rapidez del envío del paquete no es crítica pero sí importante y a pesar de que UDP no es confiable se posee la ventaja de que solo se utilizara el protocolo de comunicación en una red local.

3. Tipos de mensajes en el protocolo.

La estructura de mensaje del protocolo de comunicación del estacionamiento fue implementada de la siguiente forma.

Se creó una estructura de datos con tres campos: Un campo que almacena el tipo de operación, otro que almacena el ID y un último campo que almacena los datos que se quieran enviar como se muestra en la **figura 3.1**

Tipo de operación (uint16_t) 4 bytes	**ID (uint32_t) 8 bytes**	**Datos (uint32_t) 8 bytes**

figura 3.1

Tipo de operación es un campo que contiene un número del 0 al 7 dependiendo del tipo de mensaje. El significado de los mensajes se pueden observar en la **tabla 3.1**

Tipo de operación	Descripción
0	Paquete generado por el cliente para realizar una solicitud de entrada de un vehículo al servidor.
1	Paquete generado por el cliente para realizar una solicitud de salida de un vehículo al servidor.
2	Paquete generado por el servidor para informarle al cliente la fecha de ingreso del vehículo y su ID para que éste imprima el ticket correspondiente.
3	Paquete generado por el servidor para informarle al cliente que no hay puestos disponibles.
4	Paquete generado por el servidor para informarle al cliente el monto que debe pagar al salir.

5	Paquete generado por el servidor para informarle al cliente que hubo un error con el ID del vehículo que está ingresando o saliendo del estacionamiento.

Tabla 3.1

En el campo del ID se almacena el ID de los vehículos que ingresen al estacionamiento. Por otra parte, en el campo de los datos se almacena información que será resumida en la **tabla 3.2**

Tipo de operación	Datos
2	Los datos en un paquete cuyo tipo de operación es igual a 4 almacenan la fecha de ingreso de un vehículo.
4	Los datos en un paquete cuyo tipo de operación es igual a 6 almacena el monto que debe pagar el vehículo que va saliendo del estacionamiento.

Tabla 3.2

4. Diagrama de estados finitos del protocolo.

4.1 Diagrama de estados finitos del cliente

A continuación, en la **imagen 4.1** se mostrará el diagrama de estados finitos correspondiente al cliente del sistema.

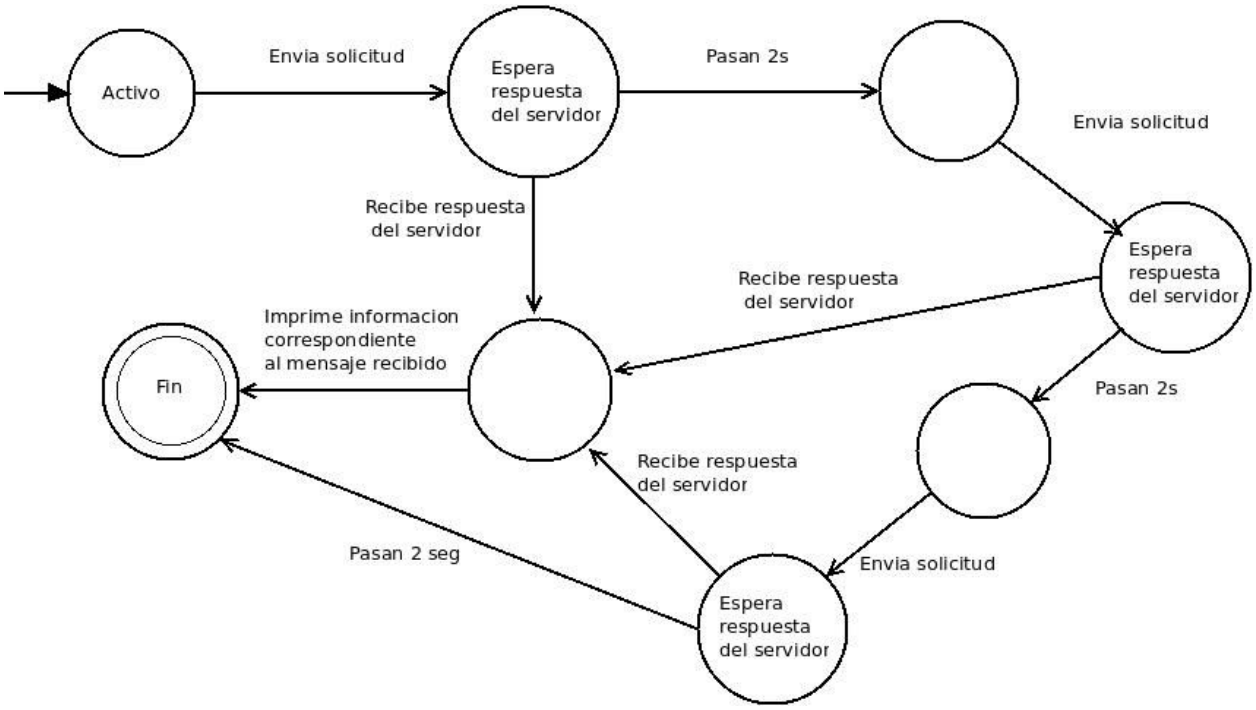

imagen 4.1

El cliente una vez activo realizara una solicitud al servidor ya sea una de entrada o salida de vehículos del estacionamiento. Si la petición es de entrada enviará un mensaje con el flag en tipo de operación encendido en 0. Si es salida entonces el flag será igual a 1. Además de esto enviará su ID en el campo ID del mensaje. Luego de enviar el mensaje espera 2 segundos por la respuesta del servidor. Si el servidor no responde en ese tiempo entonces el cliente procederá a mandar de nuevo la solicitud. Este proceso se repetirá dos veces más. Si al tercer intento si el cliente no recibe alguna respuesta del servidor entonces el mismo finalizará su sesión e informará al usuario que el servidor no está activo.

Si la recibe entonces el cliente le mostrará el ticket correspondiente al vehículo si toda los chequeos del servidor estuvieron bien o un mensaje que informe si hubieron problemas con el ID del vehículo o si no hay suficientes puestos.

4.2 Diagrama de estados finitos del servidor

En la **imagen 4.2** se puede observar el diagrama de estados finitos del servidor.

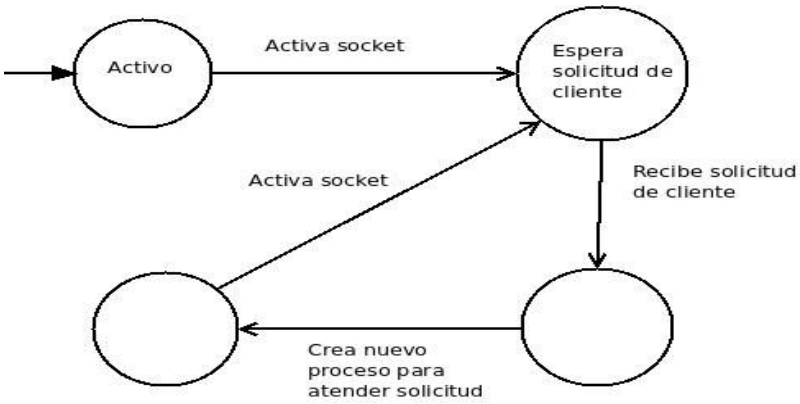

imagen 4.2

Una vez activado el servidor, la primera tarea que realiza es preparar un socket UDP para poder recibir las solicitudes realizadas por sus clientes.

Cuando el servidor recibe una solicitud de algún cliente este crea un proceso hijo. Dicho proceso será el encargado de gestionar la solicitud, armar el mensaje de respuesta y enviarlo al respectivo cliente que acudió al servidor.

Los posibles mensajes de respuesta que puede dar el servidor son:

- Mensaje cuyo campo tipo operación esté encendido en 2 para indicar al cliente que hay puesto y darle la información correspondiente sobre el ID del vehículo (guardado en el campo ID) y la fecha de ingreso del mismo (guardado en el campo datos) para que se imprima el ticket de entrada al vehículo.

- Mensaje cuyo campo tipo de operación esté encendido en 3 para indicarle al cliente que no hay puestos disponibles en el estacionamiento. Este tipo de mensajes también incluye el ID del vehículo que hizo la solicitud de entrada en el campo ID del mensaje.

- Mensaje cuyo campo tipo de operación está encendido en 4 para indicar al vehículo que va saliendo cuanto es el monto a pagar. Este tipo de mensajes también lleva en el campo ID el ID del vehículo que hizo la solicitud de salida y en el campo datos lleva el monto que se debe pagar. El cliente imprimirá el ticket correspondiente al vehículo con dichos datos.

- Mensaje cuyo campo tipo de operación está encendido en 5 para indicar al cliente que hubo problemas con el ID del vehículo bien sea porque un vehículo quería ingresar con el ID de un carro que ya estaba estacionado o porque uno quería salir con un ID de un vehículo que no había ingresado en el estacionamiento.

5. Detalles de la implementación.

- Un cliente de principio no sabrá si hay puesto o no en el estacionamiento. Este la sabrá cuando el mismo realice una petición al servidor.

Si un vehículo está entrando y hay puesto automáticamente el sistema le entregará un ticket con el número de su ID y la fecha de ingreso. Sin embargo no se le enviara informacion para decirle que hay puesto. Por el contrario, si un vehiculo quisiera entrar y no hay puesto entonces el servidor enviará un mensaje al cliente informando que no hay puestos disponibles en el estacionamiento.

- El ID que se imprime en los tickets al momento en que un vehículo desea entrar al estacionamiento será el mismo ID que el cliente ingresó al ejecutar el programa ***sem_cli*** en el flag ***-i***.

- Para calcular el monto a pagar por un vehículo una vez este quiera salir se utilizaron las siguientes tarifas :

    - La primera fracción de hora vale 80 Bs (ejemplo 0h 30min)
    - Cada hora vale 80 Bs.

- Las fracciones de hora valen 30 Bs.

Entonces, si un vehículo permanece 30 min en el estacionamiento se le cobrarán 80 Bs. Si estaciona por 2h 0 min se le cobrará 180 Bs pero si estaciona por 2h 30 min entonces el monto a pagar será de 180 Bs + 30 Bs (por la fracción de hora).

En el caso de que un vehículo permanezca 1h 59min en el estacionamiento, entonces el monto a pagar será de 110 Bs ya que en teoría no han transcurrido las 2 horas completamente.

- En el cliente se configuró un tiempo para la recepción del socket de 2 segundos. Esto quiere decir que el socket del cliente escuchará por 2 segundos y si no recibe nada intenta dos veces más. Se escogió esta cantidad de tiempo porque se consideró que configurar más tiempo sería excesivo ya que los mensajes se enviaran en una red local con poco tráfico y además el tamaño de los paquetes enviados es reducido. Por estas razones se pensó que una espera de 2 segundos era más que suficiente.

- Para almacenar la información de los carros que se encuentran estacionados se decidió que la forma más eficiente en tiempo era crear un archivo por carro el cual contenga dicha información. De esta forma se facilita la búsqueda de de un vehículo que desea salir, los chequeos de carros entrantes con id ya utilizados, la salida de carros con id no guardados y si

Guardar la información de los vehículos en archivos además de optimizar el tiempo de búsqueda evita la pérdida de información del número de carros estacionados y la información respectiva de cada uno de ellos si el servidor se llega a caer en algún momento ya que cuando se vuelva a activar el esté solo contara la cantidad de archivos que se encuentra en la carpeta donde almacena los archivos de los vehículos estacionados y asi sabra de nuevo cual es el número de puestos disponibles que posee y la información referente a estos no se habrá perdido ya que la misma no depende de una estructura.

- Ya que solo existen tres puertas (clientes) y a la simplicidad de las operaciones que el

servidor tiene que hacer, nunca va a haber un flujo de paquetes excesivo. Es por esto que la concurrencia no es una característica fundamental y por eso con que el envío de las respuestas y algunos chequeos sean hechos de forma concurrente es suficiente para que el servidor nunca se sature.

- Se utilizó una estructura de tamaño estático para los mensajes que se enviarán a través de los sockets debido a que ocupan menos espacio que las cadenas de caracteres y pueden dar toda la información que pudieran necesitar tanto el servidor como los clientes. Además de que mientras menos bits son enviados, menor es la posibilidad de que se pierda o dañe la información.

- Para los identificadores de los vehículos se decidió realizar chequeos dependiendo si era una solicitud de entrada o una solicitud de salida para evitar problemas de duplicados o de salidas de identificadores no utilizados. Es decir se revisaba la existencia de un identificador de un carro entrante para revisar si dicho identificador ya fue utilizado por un vehículo que actualmente se encuentra dentro del estacionamiento. Debido a la forma que fue implementado el sistema, aceptar vehículos con identificadores repetido causaría la pérdida de la información que se tenía del primer vehículo debido a que se sobrescribirá sus datos. Por otra parte también se revisa que el identificador de una solicitud de salida sea de alguno de los vehículos que actualmente están en el estacionamiento. De permitirse esto habrá un mal funcionamiento en el contador de los puestos disponibles, lo que haría que se aceptaran más de 200 vehículos en un mismo instante.

Conclusión

En el presente trabajo se explicó de manera detallada el diseño de un protocolo sencillo de comunicación que se utilizara en un sistema informático basado en el paradigma cliente/servidor que permitirá a su vez controlar el acceso y la salida de los vehículos que hacen uso de estacionamiento del Centro Comercial Moriah.

Para implementar este protocolo se hizo uso de sockets UDP debido a la poca información que se iba a enviar a través de la red y porque dicho protocolo solo correrá exclusivamente en la red local del estacionamiento, lo que aminora considerablemente la pérdida de los datos que serán enviados.

Se diseñó además una estructura estática con tres campos numéricos que almacenará toda la información que se desee enviar del cliente al servidor o viceversa. Los campos de dicha estructuras son : el tipo de operación, el ID y los datos. En este protocolo existen seis tipos de operación, cada tipo de operación es representada por un número que va del 0 al 5 y dependiendo del tipo la operación la información que va almacenada en el resto de los campos y el comportamiento del cliente o el servidor cambiarán. El ID del vehículo que está realizando las operaciones será un dato constante en los mensajes del cliente y el servidor.

El servidor fue implementado para que trabajara de manera concurrente. La concurrencia se implementó mediante procesos y el uso de archivos. Básicamente el trabajo del servidor será escuchar a través de su socket y cuando llegue la solicitud de alguno de sus clientes creará un hijo y este se encargará de gestionar dicha solicitud y responder.

Por otra parte, el cliente fue implementado para que fuese persistente, es decir, si no recibe la respuesta del servidor enviará otras tres solicitudes máximo.

Una vez recibida la respuesta del servidor este se encargará de interpretar e imprimirá los mensajes correspondientes al vehículo que desee entrar o salir del estacionamiento.