Universidad Simón Bolívar

CI-4835 Redes de Computadoras

Abril-Junio 2016

Asignación 3 - Sockets

Autores:

José Cipagauta 05-38040

Nicolás Mañán 06-39883

Caracas, 26 de junio de 2016

Índice general:

Introducción

Este informe tiene como propósito plantear la solución y diseño de un sistema informático basado en el paradigma Cliente/Servidor para automatizar y controlar un el estacionamiento del centro comercial Moriah.

Para la implementación del sistema se considera el uso de sockets para la comunicación entre el cliente y el servidor indicando el flujo de entrada y salida al estacionamiento, se establecerá el protocolo UDP (*User Datagram Protocol*) como capa de transporte.

Contenido

Planteamiento del problema**:**

Se busca diseñar un sistema informático que permita automatizar y controlar el Centro Comercial Moriah, el cual dispone de 200 puestos con tres puertas que permiten la entrada y salida de vehículos, dotada con expendedor de tickets y notificación de puestos libres.

Modelado como un paradigma Cliente/Servidor el cual controlará el sistema completo junto a los equipos de las puertas que permiten el acceso o salida de vehículos.

Objetivo general:

Comprender, de manera general, el funcionamiento simple de aplicaciones y servicios en redes a través del uso del paradigma Cliente/Servidor.

Objetivos específicos:

- Aplicar los conceptos dados en clases para desarrollar el diseño e implementación de un protocolo de comunicación básico.
- Comprender el uso y la programación de la interfaz de aplicaciones (API) sockets de Berkeley.

Marco teórico y tecnología:

Paradigma Cliente/Servidor: La arquitectura cliente/servidor es un modelo de aplicación distribuida en el que las tareas se reparten entre proveedores de recursos (servidores) y demandantes (clientes).

Socket: Es un concepto por el cual dos programas situados en computadoras distintas pueden intercambiar cualquier flujo de datos, generalmente de manera fiable y ordenada. Los *Sockets de Berkeley* son una interfaz de programación de aplicaciones (API), provista usualmente por el sistema operativo para la comunicación entre procesos.

UDP (*User Datagram Protocol*): Es un protocolo sencillo de nivel de transporte basado en el intercambio de datagramas que permite enviarlos sin que se haya establecido previamente una conexión.

Solución:

Para la implementación del sistema de estacionamiento del Centro Comercial Moriah enfocado en un paradigma Cliente/Servidor, se utilizan sockets con protocolo UDP, debido a ser un protocolo más sencillo de implementar, hay un intercambio escaso de mensajes entre cliente y servidor, soporta más clientes activos y nos provee la facilidad de no necesitar una conexión establecida. La API para los sockets a utilizar son los Sockets Berkeley usados en Unix, en el lenguaje de programación C, los sockets que se rigen por el protocolo UDP son conocidos como *SOCK_DGRAM*,

El funcionamiento del estacionamiento sigue las normas indicadas en el enunciado del proyecto, donde para un vehículo ingresar al mismo realiza una petición al servidor preguntando por la disponibilidad de puestos, de haber disponibilidad el servidor central envía un mensaje informando que el vehículo puede pasar e imprimiendo un ticket de entrada, en caso contrario envía un mensaje indicando que el estacionamiento está lleno.

El ticket a imprimir contiene un identificador único para cada cliente que consta de un identificador, hora y fecha de entrada al estacionamiento, a diferencia de lo que indica el enunciado, al un vehículo ingresar al estacionamiento el servidor disminuye el contador de puestos de 200 a 0 para indicar la disponibilidad de los mismos.

Al salir del estacionamiento se le genera una factura al cliente indicando el monto a cancelar según las indicaciones en el enunciado, finalmente el servidor incrementa el contador

de disponibilidad de puestos y al cierre del día genera una bitácora con las operaciones realizadas por los distintos clientes.

Adicional el cliente intenta hasta un máximo de tres veces el envío de mensajes para garantizar que se intenta entregar de las peticiones, se asegura que como nuestros paquetes son relativamente de un tamaño pequeño manejar la pérdida de información puede ser innecesaria ya que el paquete no llegaría incompleto, más bien, se envía o no, además se implementa el tiempo de espera de manera tal que sea suficiente para evitar información duplicada.

Se creó un módulo adicional al cliente y al servidor llamado *ticket* para almacenar la estructura del ticket a imprimir al entrar al estacionamiento.

El cambio de estado de ambas entidades puede verse en las figura 1 para el caso de un servidor UDP y la figura 2 para el cliente UDP.

Paquetes enviados:

E	ID	ID	ID	ID	ID	ID	ID	ID	ID	ID	ID	ID	ID	ID	ID	ID	ID
1	30 Bytes																

Donde **e** es el flag de entrada y ID es el identificador único del ticket, enviado del cliente al servidor.

S	ID	ID	ID	ID	ID	ID	ID	ID	ID	ID	ID	ID	ID	ID	ID	ID	ID
1	30 Bytes																

Donde **s** es el flag de salida y ID es el identificador único del ticket, enviado del cliente al servidor.

N	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	1 Bytes																

Paquete de rechazo en caso de no haber disponibilidad en el estacionamiento.

Conclusiones y recomendaciones:

La implementación del sistema para estacionamiento del centro comercial Moriah debe cumplir con los requerimientos solicitados y manejar de manera exitosa el continuo flujo de automóviles que ingresan al mismo, es un sistema sencillo, por lo que no debería presentar inconvenientes ante un mayor mas no excesivo número de autos que soliciten el servicio.

Aún así se recomienda.

1. Considerar una implementación utilizando el protocolo TCP para la comunicación ya que es un protocolo más seguro evitando problemas de duplicado, pérdida de información, entre otros.
2. En caso de escalabilidad del proyecto para el estacionamiento del centro comercial Moriah se recomienda implementar concurrencia con hilos o procesos debido a que un mayor flujo de autos podría entrar al mismo tiempo al estacionamiento, en nuestro caso, al solo entrar 3 carros al mismo tiempo no es necesario la implementación de estos métodos.

Anexos

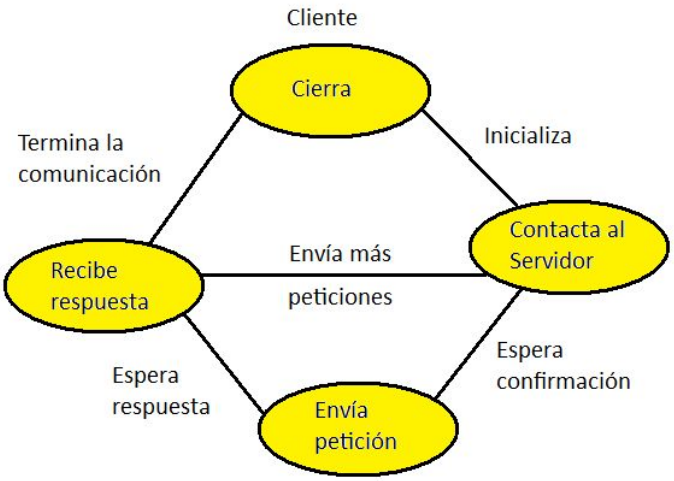

Figura 1. Diagrama de transición de estados UDP para Cliente.

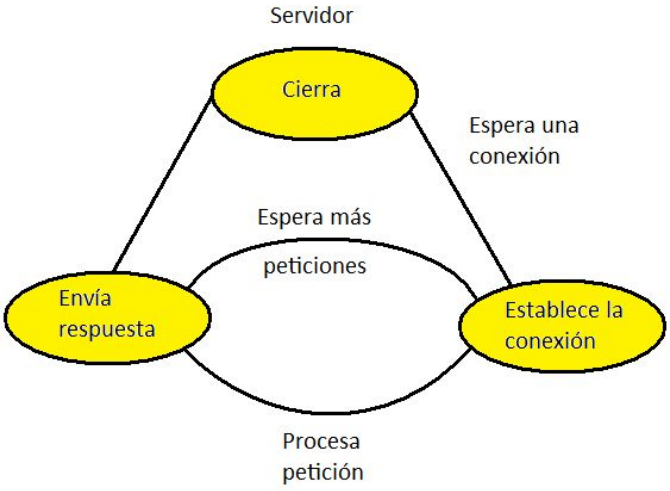

Figura 2. Diagrama de transición de estados UDP para Servidor.

Bibliografía

- Computer Networks (5ta Edicion) por Andrew S. Tanenbaum y David J. Wetherall consultado el 26/06/2016.

- Comunicaciones y Redes de Computadoras, 7ª Edición, por William Stallings. Consultado el 26/06/2016.

- https://en.wikipedia.org/wiki/Berkeley_sockets, Disponible en internet, consultado el 26/06/2016.