UNIVERSIDAD SIMÓN BOLÍVAR
DEPARTAMENTO DE COMPUTACIÓN Y TI
REDES DE COMPUTADORAS I (CI-4835)

ASIGNACIÓN 3
SISTEMA DE ESTACIONAMIENTO MORIAH
(SEM)

Integrantes:

Leonardo Martínez 11-10576

Sahid Reyes 10-10603

Sartenejas, junio de 2016

Introducción

En este documento se especifican los elementos que componen el diseño del protocolo de comunicación para el Sistema de Estacionamientos Moriah (SEM), una aplicación basada en el paradigma Cliente/Servidor e implementada en lenguaje C para el curso de Redes de Computadores 1.

Primero, se verán los aspectos generales considerados para el diseño del protocolo y tipo de socket de a utilizar. Luego se explica detalladamente el PDU del protocolo, los tipos de mensajes del sistema y los diagrama de estado correspondientes para la aplicación Cliente y la aplicación Servidor.

Finalmente se describen las consideraciones adicionales concernientes a decisiones de implementación y aspectos específicos de ésta. También se darán algunas conclusiones sobre el trabajo en completo.

Diseño del Protocolo de Comunicación

El Sistema de Estacionamiento Moriah (SEM) está constituído principalmente por un computador central (CC) y 3 puertas, haciendo una directa analogía con el paradigma Cliente/Servidor, siendo el CC el servidor y las 3 puertas los clientes.

Para la comunicación entre estos dispositivos se requiere del uso del protocolo UDP en la capa transporte, el cual es no orientado a conexión, de esa forma las peticiones del cliente hacen que los mensajes se envien rápidamente al servidor, comprometiendo la fiabilidad del mismo, por lo tanto para la definición del protocolo se han de tomar en cuenta el tipo de sockets a utilizar, el PDU del protocolo, los posibles mensajes que se puedan enviar y el funcionamiento del protocolo en sí.

Sockets a Utilizar

Para efectos de este proyecto se requiere realizar la conexión entre los clientes y el servidor por medio del protocolo UDP para la capa de transporte, escogiendo los sockets de Berkeley de datagramas no confiables, ya que no requieren de una comprobación de la conexión para enviar los mensajes por ser un protocolo no orientado a conexión, para su debida implementación se debe utilizar los atributos AF_INET para protocolos de Internet IPv4, y SOCK_DGRAM para establecer que son mensajes no confiables, sin conexión, con una longitud máxima fija, para este caso se ha considerado de 52 bytes.

PDU del Protocolo

El sistema de estacionamiento Moriah requiere de un servidor y de tres clientes que procese los mensajes provenientes entre la comunicación de los mismo, los cuales están encapsulados por un PDU único, cuyo valor en sus campos cambiarán según el tipo de mensaje ya sea por parte del cliente como del servidor. Este PDU está constituído de la siguiente forma:

Petición (4)	Tipo_paq (1)

Código (4)	Origen (1)
Puesto (1)	Placa (4)
Fecha_hora (18)	
N_ticket (4)	
Monto (4)	
Checksum (4)	

Los números en paréntesis representan el tamaño del campo en bytes.
Total de bytes: 45
Total en bytes representado realmente en C: 52 (debido al padding entre la declaraciones).

Siendo cada uno de estos campos los siguientes:

- **Petición:** entero que identifica unívocamente a cada PDU correspondiente a una petición.

- **Tipo_paq:** un caracter que representa el tipo de operación a realizar por la aplicación, ésta puede tener los valores ‘e’ para entrada del estacionamiento, ‘s’ para la salida del estacionamiento y ‘o’ para información de control.

- **Código**: entero que representa el código de informacion de control de un paquete ‘o’.

- **Origen:** un booleano que representa el origen del mensaje, siendo *false* (cero) un mensaje proveniente del cliente o *true* (uno) proveniente del servidor.

- **Puesto:** un booleano que representa la disponibilidad de los puestos del estacionamiento, siendo *false* (cero) sin puestos disponible, o *true* (uno) con puestos disponibles.

- **Placa:** entero que representa el número de serial o placa que posee el vehículo a entrar o salir del estacionamiento, considerando éste con un máximo de longitud de …

- **Fecha_hora**: siendo un string de 18 caracteres que representan la fecha y hora del sistema del servidor.

- **Num_ticket**: siento un entero unívoco para representar el código que identifica al ticket de entrada del vehículo.

- **Monto**: un entero que almacena el monto a cancelar de un vehículo al salir.

- **Checksum:** suma de comprobación para verificar la integridad de los datos del PDU.

Cada uno de estos atributos del PDU representan la cantidad necesaria de información para la comunicación entre el cliente y el servidor, ahorrando la mayor cantidad de espacio posible, siendo petición, tam_paq, origen, puesto y placa, bytes de información de control y fecha_hora, código, monto, n_ticket bytes de datos. Adicionalmente contamos con un checksum para comprobar la integridad del PDU Teniendo un total de 45 bytes.

En la implementación proporcionada se presenta la estructura del PDU con la misma disposición mostrada anteriormente. Sin embargo, por detalles de implementación en Lenguaje C y el padding entre los tipos, da un total efectivo de 52 bytes.

Con la finalidad de no saturar el socket con mucha información se ha tomado los atributos origen y puesto como booleanos, ya que al igual que los caracteres, éstos ocupan solo 1 byte, suficiente para la información que representan.

Mensajes del Sistema

Como ya se ha mencionado anteriormente, la comunicación entre el cliente y el servidor constará de un intercambio de mensajes entre ellos que dependerán de la petición del cliente (entrada o salida de un vehículo) y la respuesta del servidor ante la capacidad del estacionamiento para almacenar el vehículo. A continuación se muestran los posibles mensajes enviados en el sistema, los campos que no estén especificados no representan ningún valor relevante para el mensaje.

Provenientes del Cliente

- Mensaje de entrada al estacionamiento:
    - Orígen: Cliente.
    - Destino: Servidor.
    - Procesado por: Servidor.

Este mensaje de petición del cliente envía el tipo de operación de entrada y el serial o placa del vehículo a entrar. Contemplando los campos del PDU:

- Tipo_paq: e
- Origen: 0 (false)
- Placa: serial del vehículo indicado en la línea de comando.

- Mensaje de salida del estacionamiento:
    - Orígen: Cliente.
    - Destino: Servidor.
    - Procesado por: Servidor.

Este mensaje de petición del cliente envía el tipo de operación de salida, y el serial o placa del vehículo a salir. Contemplando los campos del PDU:

- Tipo_paq: s
- Origen: 0 (false)
- Placa: serial del vehículo indicado en la línea de comando.

Provenientes del Servidor

Para las operaciones de entrada al estacionamiento:

- Cuando el estacionamiento tiene puestos disponibles: Ticket de entrada.
  - Orígen: Servidor.
  - Destino: Cliente.
  - Procesado por: Cliente.

Después de haber recibido el mensaje de entrada al estacionamiento, el servidor calculará la cantidad de puestos disponibles, en caso de haberlo (y que no exista un vehículo con la misma placa en el estacionamiento) el servidor responderá con un mensaje de confirmación, el cual contendrá el ticket de entrada del vehículo. Contemplando los campos del PDU:

- Tipo_paq: e
- Origen: 1 (true)
- Puesto: 1 (true) / 0 (false) dependiendo si luego de la entrada quedan puestos disponibles.
- Placa: serial del vehículo contemplado en el mensaje de entrada al estacionamiento.
- Fecha_hora: fecha y hora del sistema del servidor al enviar el mensaje.
- Código: código unívoco del ticket a generar.

Para la salida del vehículo del estacionamiento:

- Salida del vehículo:
  - Orígen: Servidor.
  - Destino: Cliente.
  - Procesado por: Cliente.

Después de haber recibido el mensaje de salida del estacionamiento, el servidor calculará el monto a pagar del vehículo. Contemplando los campos del PDU:

- Tipo_paq: s
- Origen: 1 (true)
- Placa: serial del vehículo contemplado en el mensaje de salida del estacionamiento.
- Fecha_hora: la hora de salida del vehículo.
- Monto: cantidad calculada por la cantidad de horas del vehículo en el estacionamiento.

Mensajes de información provenientes del Servidor

Estos mensajes se envían cuando el servidor necesita informarle algo al cliente, los tres escenarios que se plantean son:

1) Cuando el estacionamiento no tiene puestos disponibles.
2) Cuando se intenta ingresar un vehículo que ya se encuentra en el estacionamiento.
3) Cuando se intenta retirar un vehículo que no está en el estacionamiento.

Estos tres mensajes se distinguen por el campo “código”.

- Cuando el estacionamiento no tiene puestos disponibles:
  - Orígen: Servidor.
  - Destino: Cliente.
  - Procesado por: Cliente.

Después de haber recibido el mensaje de entrada al estacionamiento, el servidor calculará la cantidad de puestos disponibles, en caso de no haber ninguno el servidor enviará un mensaje de información al cliente indicando que no hay puestos disponibles. Contemplando los campos del PDU:

  - Tipo_paq: o
  - Origen: 1 (true)
  - Codigo: 0
  - Puesto: 0 (false)

- Cuando el vehículo que quiere ingresar ya se encuentra en el estacionamiento:
  - Orígen: Servidor.
  - Destino: Cliente.
  - Procesado por: Cliente.

Después de haber recibido el mensaje de entrada al estacionamiento y existen puestos disponibles el servidor busca en el estacionamiento la placa del vehículo que quiere ingresar, si esta ya se encuentra registrada, entonces se envía un mensaje indicando al cliente que ya existe un vehiculo con esa placa en el estacionamiento. Contemplando los campos del PDU:

  - Tipo_paq: o
  - Origen: 1 (true)
  - Codigo: 1
  - Puesto: 0 (false)

- Cuando el vehículo que quiere abandonar el estacionamiento pero este no se encuentra registrado en el mismo:
  - Orígen: Servidor.
  - Destino: Cliente.
  - Procesado por: Cliente.

Después de haber recibido el mensaje de salida del estacionamiento el servidor busca en el estacionamiento la placa del vehículo que quiere salir, si esta no se encuentra registrada, entonces se envía un mensaje indicando al cliente que no existe un vehiculo con esa placa en el estacionamiento. Contemplando los campos del PDU:

- Tipo_paq: o
- Origen: 1 (true)
- Codigo: 2
- Puesto: 0 (false)

Para garantizar la fiabilidad de la aplicación, se ha tomado la decisión de que el servidor siempre envía mensajes de confirmación al cliente, ya sea un mensaje con la respuesta de la petición o un mensaje de información.

Comunicación en el Protocolo

Para la comunicación del protocolo se enviarán los mensajes mencionados en el apartado anterior según el evento que pueda ocurrir, para el caso del Sistema de Estacionamiento Moriah se presentan dos casos generales:

- **Entrada al estacionamiento:**

1) El cliente realiza la petición al servidor, enviando un mensaje que posee una operación de entrada, y el serial o placa del vehículo a entrar.

2) Una vez enviado el mensaje, el cliente queda a la espera del ticket de estacionamiento en caso de poder entrar o en su defecto un mensaje por parte del servidor indicando que la operacion no puede ser posible.

3) En caso de tener respuesta del servidor con el mensaje de Ticket de entrada, el cliente muestra el ticket y finaliza, en caso de no recibir algún tipo de respuesta, éste sigue en modo de espera hasta que expire un temporizador de 3 segundos. Al expirar el temporizador se reenvía el mensaje de entrada al estacionamiento, esto último un máximo de 3 veces. Si aún no se obtiene respuesta, finaliza.

4) Para el servidor, al momento de ser recibido el mensaje de entrada al estacionamiento, el cliente se encuentra en modo de espera, mientras tanto el servidor realiza las operaciones necesarias para verificar que pueda ser posible la entrada, respondiendo al cliente con un mensaje de Ticket de entrada o en su defecto un mensaje de información ‘o’.

5) Para el momento en que el servidor envía su mensaje de confirmación a la puerta de la petición, éste regresa al estado de espera por una nueva petición por parte de algún cliente.

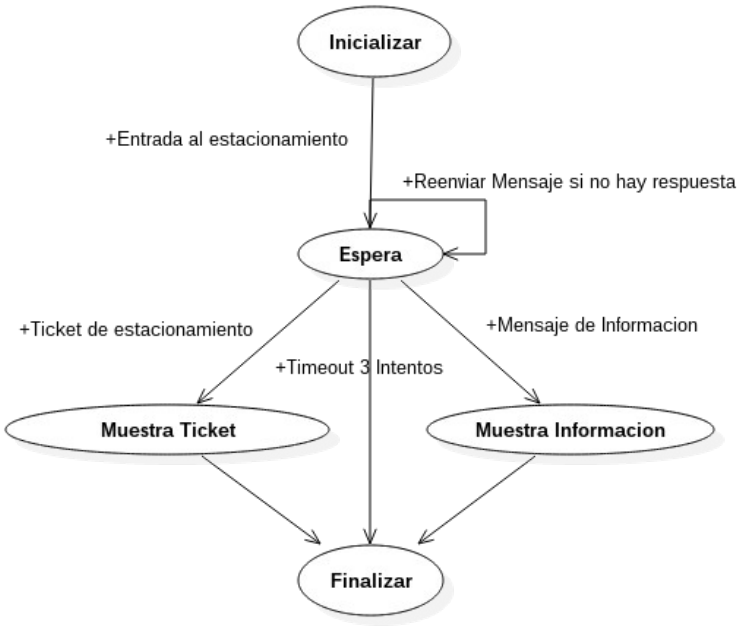

Figura 1.1 Diagrama de estados del cliente para operaciones de entrada.

- **Salida del Estacionamiento:**

1) El cliente realiza la petición al servidor, enviando un mensaje de salida del estacionamiento y el serial o placa del vehículo a salir.

2) Una vez enviado el mensaje, el cliente queda en stand-by, esperando el monto a pagar del estacionamiento, o en su defecto un mensaje por parte del servidor indicando que la operación no pueda ser posible.

3) En caso de tener respuesta del servidor con el mensaje de Salida del estacionamiento, el cliente muestra el monto a pagar y finaliza. En caso de no recibir respuesta se sigue como en el punto 3 del procedimiento de entrada.

4) Para el servidor, al momento de ser recibido el mensaje de salida del estacionamiento, el cliente se encuentra en modo de espera, mientras tanto el servidor realiza las operaciones necesarias para calcular el monto a pagar y verificar que pueda ser posible la salida, respondiendo al cliente con un mensaje de Salida del estacionamiento o un mensaje de información ‘o’.

5) Para el momento en que el servidor envía su mensaje de confirmación a la puerta de la petición, éste regresa al estado de espera por una nueva petición por parte de algún cliente.

Figura 1.2 Diagrama de estados del cliente para operaciones de salida.

- **Estados del servidor:**

1) Una vez el servidor se inicializa, queda constantemente en estado de espera por peticiones de los clientes.

2) Cuando se recibe una petición del cliente (entrada o salida) se procesa y se envia respuesta apropiada al servidor. Si llega una segunda petición mientras el servidor está atendiendo una petición ya hecha, esta se encola hasta que pueda ser atendida. Dichas peticiones se atenderán de manera secuencial. Peticiones adicionales mientras la cola esté a su máxima capacidad serán ignoradas.

3) Una vez finalizada la petición, el servidor vuelve al estado de espera.

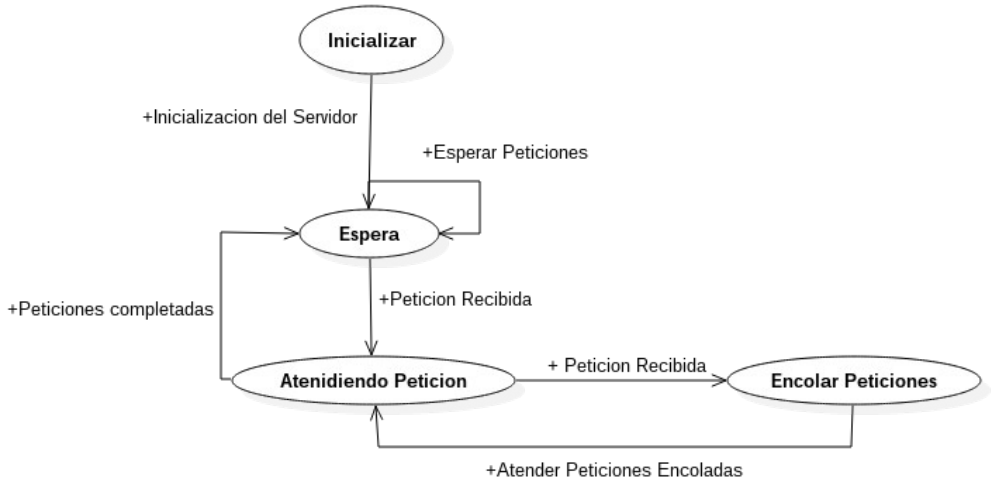

Figura 1.3 Diagrama de estados del Servidor.

Consideraciones Adicionales

Las peticiones se atienden de manera secuencial. El servidor no trabaja de manera concurrente (hilos o procesos), considerando que es un desperdicio de procesamiento para la cantidad de puertas a atender (solamente 3). En caso de existir peticiones simultáneas que generen colisión y corrupción en los datos, el servidor descartará el paquete corrupto, se confía que el cliente siguiendo el protocolo enviará nuevamente su petición al no recibir respuesta.

Considerando los puntos opcionales del enunciado del proyecto, se han hecho estrategias dentro de la capa de aplicación y el protocolo para manejar de la mejor manera posible la información duplicada, evitar reprocesamiento de peticiones ya atendidas, pérdida de paquetes y garantizar la integridad de la información.

Para la integridad de la información:

Se ha añadido un campo checksum que corresponde a la suma de comprobación aplicada al PDU del sistema SEM. Tanto el cliente como el servidor realizan comprobación del checksum de los paquetes entrantes y agrega el checksum a sus paquetes salientes.

¿Qué ocurre cuando un paquete de entrada no coincide con su checksum?

- **Del lado del servidor:** Se ignora el paquete y se confía que el cliente volverá a emitir la petición al no recibir respuesta.

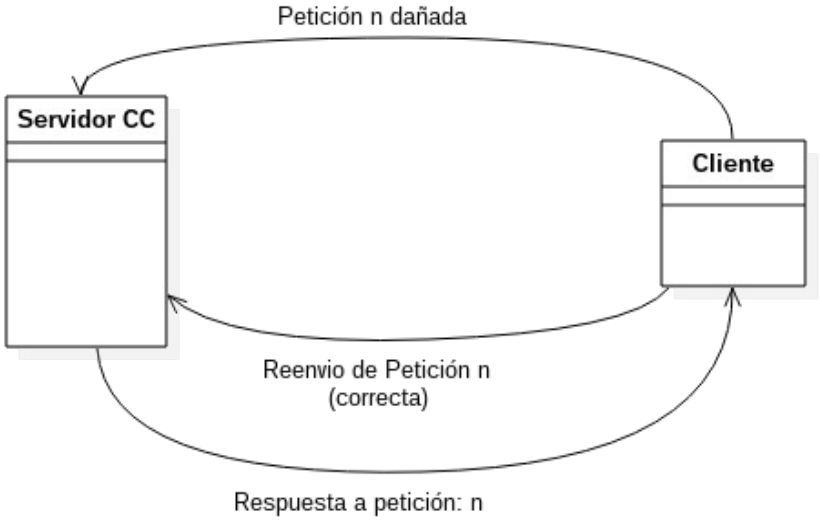

Figura 2.1 Comprobación del checksum desde el servidor

- **De lado del cliente:** Se vuelve a enviar un mensaje con la misma petición, se confía que el servidor sabe que la petición ya fue atendida y este enviará una copia de la respuesta original.

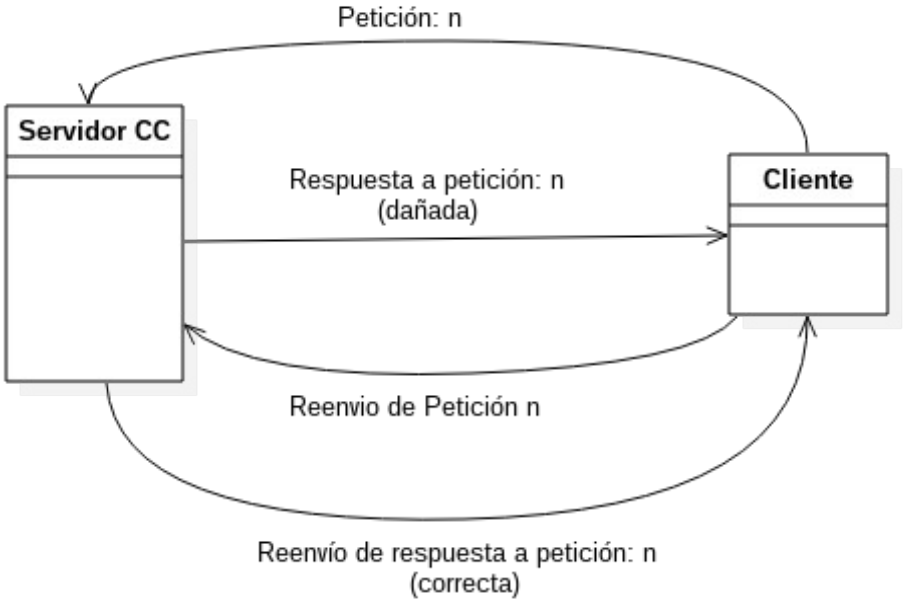

Figura 2.2 Comprobación del checksum desde el cliente

Para la duplicidad y el reprocesamiento de peticiones:

Aquí tiene protagonismo el campo **petición** del PDU. El servidor cuenta con un registro de las últimas 20 peticiones respondidas. Suponiendo que en algún momento se pierda una respuesta del servidor al cliente, se sabe que el cliente eventualmente volverá a solicitar la misma petición, de recibirse una petición que ya fue atendida, esta no se reprocesa y se reenviará una copia de la respuesta enviada previamente.

La cantidad de veces en que se reenvía la respuesta al cliente dependerá de la cantidad de veces que éste envíe de nuevo la información de la petición. Considerando la configuración actual del protocolo, esto es un máximo de 2 reenvíos, una vez que finalice el último reenvío, no se garantiza que el cliente reciba la información de respuesta, implicando que el cliente deberá realizar una nueva petición al servidor que generará una nueva respuesta.

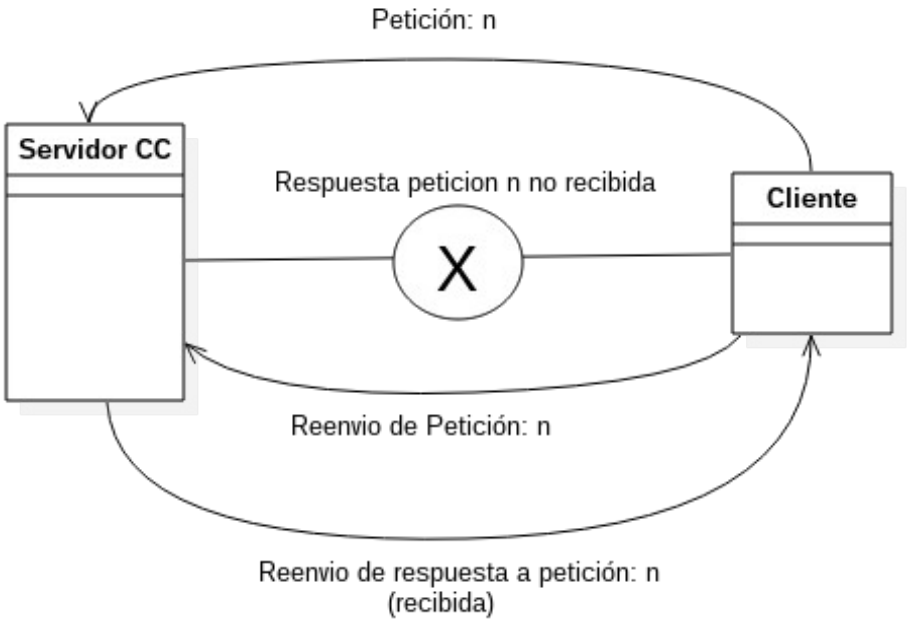

Figura 2.3 Duplicidad de peticiones.

Conclusión

El diseño del protocolo de comunicación del Sistema de Estacionamientos Moriah (SEM) ha permitido poner en práctica los conocimientos adquiridos en la teoría de la asignatura Redes de Computadoras I mediante la estructuración de la comunicación para una aplicación Cliente/Servidor.

Durante la implementación del protocolo, se pudo apreciar el funcionamiento del protocolo UDP y como trabaja encapsulando los datos a enviar en capa de red, asi como las diferentes estrategias a implementar en capa de aplicación para hacer este protocolo un poco más confiable y que no solo recaiga en la filosofía del “mejor esfuerzo”.

Finalmente se pudo aprender, durante la formulación de los casos hipotéticos del funcionamiento del sistema, sobre la importancia que puede tener un protocolo de comunicación en un modelo de negocio, como el caso del Estacionamiento Moriah, donde pérdidas de paquetes, corrupción de datos y latencia en la entrega de mensajes pudiesen ocasionar impacto en el servicio prestado y provocar pérdidas económicas.

Referencias

- The GNU reference manual. https://www.gnu.org/software/gnu-c-manual/gnu-c-manual.html
- Comer D, Stevens D. Internetworking with TCP/IP, Volume 3: Client-Server Programming and Applications.