Proyecto: Vehículo Autónomo de Telemetría

Asignatura: Internet Arquitectura y Protocolos

Porcentaje: 20%
2025-2

Introducción

En este primer proyecto de curso se pondrán a prueba sus habilidades para la programación de aplicaciones de red, que soportan concurrencia, así como sus competencias para diseñar e implementar protocolos de la capa de aplicación.

En este curso, un protocolo es simplemente el conjunto de reglas para que dos programas se comuniquen. Dichas reglas incluyen los formatos de mensajes y procedimientos que se generan que controlan el intercambio coordinado de información y de servicios entre entidades que se conectan a una red de datos.

En este caso, un vehículo autónomo terrestre envía información de telemetría a múltiples usuarios y recibe comandos de control. Se usará la API de Sockets Berkeley para crear un servidor en C y clientes en al menos 2 lenguajes distintos.

Conceptos

Sockets y Abstracción

Un socket es una abstracción que permite a las aplicaciones comunicarse a través de una red. Una aplicación escribe datos en su socket y otra los lee desde el suyo, incluso si se encuentran en diferentes máquinas.

Tipos de Sockets en el Proyecto

Existen diferentes tipos de sockets, pero para este proyecto se consideran dos de la API de Berkeley:

- Sockets de flujo (SOCK_STREAM): Proporcionan una conexión fiable, ordenada y orientada a la conexión. Son ideales para aplicaciones que no pueden permitirse la pérdida o el desorden de datos.
- Sockets de datagrama (SOCK_DGRAM): Ofrecen un servicio más rápido y sin conexión. No garantizan la fiabilidad ni el orden de los datos, lo que los hace adecuados para aplicaciones donde la velocidad es más importante que la precisión.

Para este proyecto debe decidir y justificar en qué escenarios usar cada uno.

Contexto

Un equipo de ingenieros ha desarrollado un vehículo autónomo que mide y transmite variables como velocidad, nivel de batería y temperatura interna. Además, puede recibir instrucciones para modificar su movimiento o velocidad. Usted ha sido asignado para diseñar e implementar el protocolo de comunicaciones que permita a varios usuarios obtener las mediciones en tiempo real y enviar comandos de control.

Requerimientos

1. El vehículo debe enviar información de telemetría a todos los usuarios conectados cada 10 segundos. En este sentido, la aplicación en el robot debe mantener una lista actualizada, añadiendo o removiendo usuarios según el estado de conexión.
2. Los comandos que puede recibir son: SPEED UP, SLOW DOWN, TURN LEFT, TURN RIGHT. El vehículo puede responder que no ejecuta el comando (por ejemplo, por batería baja o límite de velocidad).
3. Existen dos tipos de usuarios: Administrador (envía comandos y consulta usuarios) y Observador (solo recibe datos). Debe garantizarse la identificación y autenticación del administrador incluso si este cambia de IP.
4. Diseñar, especificar e implementar el protocolo de la capa de aplicación en formato texto. Debe incluir: especificación del servicio, vocabulario de mensajes, formato de encabezados, reglas de procedimiento y todo lo que usted considere necesario para el buen funcionamiento del protocolo. Para esto se recomienda considerar la estructura, por ejemplo, de un RFC como el 793 (TCP), que, a grandes rasgos, incluye:
    a. Visión General del Protocolo: Se define el propósito del protocolo, su modelo de funcionamiento (por ejemplo, cliente-servidor) y la capa de la arquitectura en la que opera.
    b. Especificación del Servicio: Se describen las primitivas del servicio, que son las operaciones que un usuario del protocolo puede realizar (por ejemplo, GET, MOVE, LIST).
    c. Formato de Mensajes (Encabezado): Se desglosa la estructura del encabezado del protocolo. En este punto, se describen los campos del encabezado, su tamaño, su significado y los valores posibles (por ejemplo, parámetros solicitados, longitud del movimiento, entre otros).
    d. Reglas de Procedimiento (Flujo de Operación): Se detallan los algoritmos y las máquinas de estado para el funcionamiento del protocolo.
    e. Ejemplos de Implementación: Se pueden incluir ejemplos de código o diagramas de secuencia para ilustrar el funcionamiento del protocolo en escenarios específicos.

5. El protocolo debe integrarse a TCP/IP mediante la API de sockets y decidir el protocolo de transporte adecuado.
6. El sistema debe soportar múltiples clientes simultáneos mediante el uso de hilos para garantizar la concurrencia en el servidor.
7. Debe implementar control de excepciones mediante el procedimiento para conexiones fallidas, mensajes inválidos, etc.

Detalles de la implementación

- Servidor programado en lenguaje C usando API de Berkeley, no se admite ninguna otra.
- Clientes programados en al menos 2 lenguajes distintos (ej. Python y Java).
- El cliente debe tener una interfaz gráfica sencilla que muestre velocidad, batería y dirección en tiempo real.
- El servidor debe ejecutar una funcionalidad de logging, mostrando en consola y guardando en archivo las peticiones y respuestas.
- El servidor se debe ejecutar con parámetros: ./server <port> <LogsFile>. Identificando al puerto por el que recibe peticiones y al archivo donde almacena las peticiones y respuesta. Las peticiones y respuestas deben almacenar el identificador de cada cliente (ip y puerto origen de la petición).
- Se recomienda usar gcc y un Makefile para compilar.
- Equipo de trabajo: El proyecto debe ser desarrollado en grupos de 3 o 4 personas como máximo. NO debe ser desarrollado de manera individual.
- **Fecha de entrega**: El proyecto se debe culminar el 5 de octubre de 2025 hasta las 23:59.
- **Mecanismo de entrega**: Por el buzón de Interactiva virtual se debe realizar la entrega.
- La entrega debe incluir un enlace a un repositorio en github.
- Incluir un enlace a un video (de 12 min aprox.) donde se explique el código y se muestre el funcionamiento.
- A partir de esta fecha y hora, no se puede realizar ningún commit al repositorio.

Evaluación

La calificación del proyecto dependerá de varios componentes. Cada uno de ellos será evaluado de manera presencial, sustentando lo desarrollado de forma oral. En la sustentación cada estudiante debe ser capaz de explicar de forma clara, precisa y concisa el funcionamiento de su código y las decisiones de diseño. También debe poder responder preguntas sobre aspectos técnicos, teóricos y de implementación del proyecto. Los aspectos por evaluar comprenden:

1. Programación del Servidor (40%)

• Implementación de la API de Sockets (5%): El servidor utiliza correctamente la API de Sockets Berkeley para crear, enlazar y poner a la escucha un socket.

• Manejo de Múltiples Clientes (5%): El servidor puede aceptar y gestionar múltiples peticiones de clientes de forma concurrente, sin bloquearse

• Lógica del Protocolo (30%): El servidor implementa de forma correcta el protocolo diseñado, procesando los mensajes según las reglas establecidas.

2. Programación de los Clientes (30%)

• Conexión y Petición (15%): Los clientes se conectan correctamente al servidor y construyen las peticiones según el formato del protocolo.

• Manejo de Respuestas (15%): Los clientes procesan y muestran adecuadamente las respuestas del servidor, incluyendo mensajes de éxito y de error.

3. Funcionamiento y Comunicación (20%)

• Robustez del Protocolo (10%): El sistema completo (servidor y clientes) funciona de manera estable ante diferentes escenarios (ej. múltiples clientes, peticiones inválidas, desconexiones).

• Coordinación de Mensajes (10%): El intercambio de mensajes entre las entidades es coordinado y cumple con las reglas de procedimiento del protocolo.

4. Documentación e Informe (10%)

• Especificación del Protocolo (10%): informe en formato APA donde se describe de forma detallada la especificación del protocolo, incluyendo el formato de los mensajes y las reglas de procedimiento.

Referencias

a) https://man7.org/linux/man-pages/man7/socket.7.html
b) https://datatracker.ietf.org/doc/rfc2616/
c) https://www.wireshark.org/
d) https://www.postman.com/
e) https://www.geeksforgeeks.org/tcp-server-client-implementation-in-c/
f) https://beej.us/guide/bgnet/