Droneswarms

1. Arquitectura General del Sistema

El punto de entrada del sistema es el **programa** `main`, el cual se encarga de inicializar todo el entorno de simulación. Desde este proceso principal se crean e invocan múltiples procesos independientes, entre los cuales destacan:

- **Truck (camión):** representa un vehículo terrestre encargado de desplegar drones.
- **Enemy (defensa enemiga):** simula posibles amenazas en la zona de operación.
- **Targets (objetivos):** entidades que poseen una **posición fija** y un **identificador único (ID)**, definidos desde el inicio para asignarlos a drones. Pero no son un proceso.

2. Procesos Truck y Drones

Cada proceso **truck** actúa como una estación de lanzamiento y control de drones. Al iniciar, cada camión crea uno o varios procesos **drone**, los cuales pueden ser de dos tipos:

- **Drone de tipo CÁMARA:** se encarga de recolectar información visual sobre el entorno.
- **Drone de tipo ATTACK:** su función es neutralizar objetivos específicos.

Inicialmente, cuando un drone es creado, permanece en estado de **espera**. El camión que lo lanzó utiliza **sockets** para enviarle información sobre su objetivo, incluyendo su posición y el identificador asignado. Este mecanismo de comunicación entre procesos permite que cada drone conozca de antemano cuál será su misión.

3. El Proceso Comando y Asignación de Drones

Un proceso especial, llamado **comando**, cumple el rol de **centro de control principal** para todos los drones del sistema. Este proceso es creado por un truck especial y se encarga de:

1. **Administrar la asignación de drones:** lleva un registro de qué drones están disponibles y cuáles tienen un objetivo asignado.
2. **Controlar el estado de vuelo:** indica a cada drone si debe despegar, mantenerse en vuelo circular o dirigirse a un objetivo.
3. **Coordinar la comunicación:** establece canales de comunicación dedicados con cada drone.

Para lograr una comunicación eficiente, **comando** crea **múltiples hilos de ejecución**. Cada hilo se encarga exclusivamente de manejar el socket de un drone en particular. Esto permite que varios drones puedan intercambiar datos con **comando** de forma paralela, evitando bloqueos y maximizando la velocidad de respuesta.

4. Uso de Listas Enlazadas para la Gestión de Enjambres

Para almacenar información sobre los drones y sus respectivos enjambres, **comando** utiliza una **lista doblemente enlazada** declarada únicamente con macros (descargada de la librería **sys/queue.h** de FreeBSD) como estructura de datos principal. Cada nodo de la lista representa un enjambre específico y contiene información sobre los drones que lo conforman, los cuales también están en una lista doblemente enlazada.

La principal ventaja es la eficiencia en operaciones dinámicas**,** al no requerir un espacio contiguo, la inserción y eliminación son rápidas y se realizan en **O(1)**, siempre que se tenga la referencia al nodo.

5. Sincronización mediante Semáforos

Uno de los aspectos clave en el diseño del drone es el uso de **semáforos** para controlar las transiciones de estado durante la misión. Cada hilo encargado del sistema de vuelo permanece bloqueado en un semáforo, a la espera de una señal que solo puede ser emitida por el **sistema network**.

- Si **comando** ordena al drone **iniciar el vuelo**, el sistema network libera el semáforo correspondiente, activando al hilo de vuelo.
- Si la orden es **volar en círculos**, otro semáforo se desbloquea para que el hilo ejecute la maniobra.
- Finalmente, cuando **comando** le indica que **avance hacia el objetivo**, se activa un tercer semáforo que permite iniciar la trayectoria final.

6. Comunicación entre Procesos

La comunicación entre todas las entidades del sistema se realiza mediante **sockets**. El diseño incluye dos tipos principales de comunicación:

1. **Truck → Drone:** para asignar objetivos y datos iniciales.
2. **Drone ↔ Comando:** para enviar actualizaciones de estado y recibir nuevas instrucciones.