Real-Time Radar Data Processing System

Thursday 20th March, 2025

Problem Description

You are tasked with implementing a system that simulates a radar system for detecting and tracking objects. The system consists of four modules that work together to process radar data in real-time. The system must meet the following requirements:

Module 1: Radar Data Generator

- Simulates a radar system that continuously generates **radar pulses**.
- Each radar pulse is represented as a data structure containing:
  - A unique pulse ID (UUID).
  - A UTC timestamp indicating when the pulse was generated.
  - A list of **detected objects**, where each object is represented by:
    * Range (distance from the radar in meters).
    * Azimuth (angle in degrees relative to the radar).
    * Velocity (speed in meters per second).
    * Signal strength (in dBm).
- The number of detected objects in each pulse is random [0...50].
- The radar pulses are generated at a rate of **10 pulses per second**.
- The generated pulses are delivered to **Module 2** via a shared buffer with a capacity of 1000 pulses.

Module 2: Radar Data Filter

- Receives radar pulses from **Module 1** and filters them based on the following criteria:
  - Only pulses containing objects with a **signal strength greater than -90 dBm** are forwarded to **Module 3**.
  - Pulses that do not meet the signal strength criteria are discarded.
- The module uses an output buffer with a capacity of 500 pulses to deliver filtered data to **Module 3**.

Module 3: Object Tracker

- Receives filtered radar pulses from **Module 2**.
- Tracks detected objects over time using a **Kalman filter** or a simple moving average algorithm.
- For each object, the module maintains:
  - A unique object ID (UUID).
  - A history of its range, azimuth, velocity, and signal strength over time.

- A confidence score indicating the likelihood that the object is a valid target (e.g., not noise).

- The module updates the object tracks every second and forwards the updated tracks to **Module 4** via a shared buffer with a capacity of 1000 tracks.

Module 4: Radar Data Visualizer

- Receives object tracks from **Module 3**.
- Simulates a radar display by printing the following information to the console every second:
  - A list of all tracked objects, sorted by range.
  - For each object, display:
    - Object ID.
    - Current range, azimuth, velocity, and signal strength.
    - Confidence score.
- The module also writes the tracked object data to a log file (`radar_tracks.log`) in CSV format for later analysis.

Additional Requirements

- **Thread Safety:**
  - All shared data structures (e.g., buffers, queues) must be thread-safe.
  - Use appropriate synchronization primitives (e.g., mutexes, condition variables) to avoid race conditions.
- **Real-Time Processing:**
  - Ensure that the system can handle the high data rate of 10 pulses per second without significant delays.
- **Modularity:**
  - All modules must implement the same interface (`IModule`), which includes methods for starting, stopping, and configuring the module.
  - Each module must run in its own internal thread.

Main Application Pseudo-Code

```
int main(void) {
    // Create modules
    IModule *m1 = (get object of type RadarDataGenerator);
    IModule *m2 = (get object of type RadarDataFilter);
    IModule *m3 = (get object of type ObjectTracker);
    IModule *m4 = (get object of type RadarDataVisualizer);

    // Set up module connections
    m1->setOutputModule(m2);
    m2->setOutputModule(m3);
    m3->setOutputModule(m4);

    // Start modules
    m1->start();
    m2->start();
    m3->start();
    m4->start();

    // Let the system run for 300 seconds
    std::this_thread::sleep_for(std::chrono::seconds(300));

```

```
    // Stop modules
    m1->stop();
    m2->stop();
    m3->stop();
    m4->stop();

    // Release modules
    delete m1;
    delete m2;
    delete m3;
    delete m4;

    return 0;
}
```

Key Challenges

- **Radar Data Simulation:**
  - Simulate realistic radar pulses with random objects, ranges, azimuths, velocities, and signal strengths.
- **Object Tracking:**
  - Implement a tracking algorithm (e.g., Kalman filter) to maintain object tracks over time.
- **Real-Time Performance:**
  - Ensure that the system can process 10 pulses per second without dropping data or introducing significant delays.
- **Data Visualization:**
  - Simulate a radar display by printing object tracks to the console in a readable format.
- **File I/O:**
  - Write tracked object data to a log file in CSV format for later analysis.

Expected Output

- **Console Output (Radar Display):**
  - Every second, print a list of tracked objects sorted by range. Example:

```
[UTC Timestamp] Tracked Objects:
- Object ID: XYZ123, Range: 1200m, Azimuth: 45\textdegree, Velocity: 250m/s,
   Signal: -80dBm, Confidence: 0.95
- Object ID: ABC456, Range: 3500m, Azimuth: 120\textdegree, Velocity: 150m/s,
   Signal: -85dBm, Confidence: 0.90

```

- **Log File Output (`radar_tracks.log`):**
  - Append tracked object data to the log file in CSV format. Example:

```
UTC Timestamp,Object ID,Range (m),Azimuth (\textdegree),Velocity (m/s),Signal (
   dBm),Confidence
2023-10-01T12:00:00Z,XYZ123,1200,45,250,-80,0.95
2023-10-01T12:00:00Z,ABC456,3500,120,150,-85,0.90

```

Bonus Challenges

- Add a **Module 5** that analyzes the tracked objects and identifies potential threats (e.g., objects with high velocity and low range).
- Implement a mechanism to dynamically adjust the radar pulse rate based on the number of detected objects.
- Add support for multiple radar systems (e.g., multiple instances of **Module 1**) and merge their data in **Module 3**.