Background

One of the core advantages of the Ommo system is its ability to support a linearly scalable number of devices that are simultaneously tracked. Each device generates data in real-time based on its hardware and software configuration to be processed by the algorithm, and the processed data is then made available to the end customer clients as requested. Given the amount and frequency of the data, it is critical that the data flow throughout Ommo's system is as efficient as possible.

In this exercise, we'll explore implementing a simplified version of the data routing system and measuring its performance.

Data

To start, we'll consider the data generated by the hardware.

Each hardware generates a sample of data at a set interval.
The sample packet of data contains the following information:

- UUID - the unique identifier for the hardware device
- Timestamp - the timestamp that the data for this packet is sampled at
- Magnetic samples - samples of the magnetic sensors
    - The samples depend on the number of magnetic sensors this hardware has
    - For each magnetic sensor, the sample would contain 3 floats representing x,y,z reading of the sensor
        - For example, a hardware device with 3 magnetic sensors will have 9 float values for each sample packet
- Accelerometer samples - samples of accelerometer sensors
    - Same as magnetometer where each hardware can have a number of accelerometer sensors, and each sensor would produce 3 float values per sample
    - The number of accelerometers will always be equal to or less than the number of magnetometer sensors
- Gyroscope samples - samples of the gyroscope sensors
    - Same as magnetometer where each hardware can have a number of gyroscope sensors, and each sensor would produce 3 float values per sample
    - The number of gyroscopes will always be equal to the number of accelerometer sensors

Data Flow

The flow of data is as follows:

- Each hardware device generates a sample packet of data at a set interval based on its configuration
    - The interval can be 1ms, 0.5ms, or 0.33ms
- The hardware samples will be sent to the algorithm for processing

    - An algorithm instance will be processing data for ONE individual magnetometer
        - This means that a hardware device with multiple magnetometers will need a corresponding number of algorithm instances
        - The processing time can vary. The time will generally be about half of the sampling interval, but it can go up to 2x the sampling interval occasionally
- Once all of the algorithm instances are done processing the sample packet, the resulting poses are computed
    - Each pose consists of 7 floating point values.
    - Each magnetometer (algorithm instance) will produce one pose
- The sample packet with its raw data and its associated pose outputs will be sent to any downstream clients that have requested data from this particular device

Requirements & Objectives

1. Implement the data flow described above as the data routing system
    a. Each hardware must be generating sample packets running in its on thread
        i. You can simulate producing the samples per hardware at specified intervals. The data can be anything but the UUID must be unique.
        ii. You must use at least 10 hardware devices (10 threads)
        iii. The hardware devices must include a range of configurations with different number of sensors of each type ranging from 1 to 5
    b. Algorithm processing time must be accounted for with a proper distribution as described in the data flow
        i. Processing time can be simulated by simply blocking/sleeping for the duration of “algorithm work”
    c. There must be at least 5 downstream “clients” consuming the resulting samples with pose outputs
        i. The clients must consume from at least one hardware device
        ii. There must be clients that consumer from at least three hardware devices, and those devices must also be consumed by at least two other clients
    d. You have to decide on how to gracefully handle congestions when algorithm processing takes longer than the sample interval
    e. You must demonstrate proper concurrency and thread safety where necessary
    f. For each hardware device, its samples must be guaranteed to be received by downstream consumers in the order that they were produce
        i. The order of samples across hardware devices do not have to be maintained
    g. [Bonus] Demonstrate use of lockless concurrency if and where appropriate
2. Once the data flow is implemented, you’ll implement a method or methods of your choice to measure the performance of your data flow implementation.
    a. The measurement method must produce numbers that show the rate at which samples are produced, processed, and consumed
    b. Any samples dropped due to unable to be processed in time should be counted

c. Be prepared to explain how you'll improve your design and what issues you discovered based on your measurement method
d. Be prepared to discuss any shortcomings and caveats with your measurement method

C++ is the preferred language but you are free to use any language of your choice and any external 3rd party libraries. However, build/running instructions must be provided, including for any dependencies if you use them.
We MUST be able to run your code to generate the output metrics from your performance measurement method.