SWAVLAMBAN 2025 HACKATHON CHALLENGE - 1

DEVELOPMENT OF DISTRIBUTED SWARM ALGORITHM

1. **Core Technical Objectives.** Participants must build a single Python agent file or Docker container that implements a distributed swarm algorithm. The algorithm must address three main challenges:

1.1 **Leader Election.** The swarm must be able to detect when its current leader fails (e.g., by missing periodic beacon messages) and elect a new leader within **10 seconds**.

1.2 **Capability-Aware Task Allocation.** The system must assign various pop-up tasks to the most suitable vehicle. This decision should be based on factors like the vehicle's specific capabilities (e.g., sensors, tools), its proximity to the task, and task deadlines. The task allocation must stabilize within **60 seconds** after any changes in the swarm.

1.3 **Disciplined Communications.** The algorithm must operate over lossy, low-bandwidth radio links (capped at 64 kbps per node). It should use event-triggered, short messages to minimize network usage.

2. **Simulation and Scenarios.** The challenge does not require any real-world Navy data. Instead, participants will use a provided deterministic simulator called **SWARMBENCH-30+** and test their solutions against seven fixed benchmark scenarios (S1-S7).

2.1 **S1-S5.** These are baseline scenarios that vary in communication quality (bandwidth, data loss) and task density.

2.2 **S6 (Leader-down).** This scenario specifically tests the leader election capability. The designated leader vehicle fails at the 300-second mark.

2.3 **S7 (Capabilities).** This tests task allocation where tasks require specific capabilities, and each vehicle in the swarm has a different set of capabilities. A task is only successfully completed if the assigned vehicle has the matching capability.

3. **Evaluation and Scoring.** Submissions are evaluated based on a primary score and several pass/fail gates.

3.1 **Primary Score.** A single, reproducible score is calculated using the following formula:

$$\text{Score} = 0.6 \times \text{value\_ratio} - 0.2 \times \text{normalized\_distance} - 0.2 \times \text{normalized\_bytes}$$

- **Value_ratio.** The fraction of the total possible task value that the swarm successfully achieved.
- **Normalized_distance.** A proxy for fuel efficiency, penalizing excessive movement.
- **Normalized_bytes.** A measure of communication efficiency, penalizing heavy data usage.

3.2 **Pass/Fail Gates.** To qualify, a solution must meet these strict criteria:-

3.2.1 It must be a **fully distributed** system with no central controller.

3.2.2 Task ownership must be at least **95% stable within 60 seconds** of a change.

3.2.3 Leader re-election in scenario S6 must be completed in **under 10 seconds**.

3.2.4 The algorithm must run at a minimum of **10 Hz** on a standard laptop.

4. **Final Deliverables.** Participating teams are expected to deliver a complete package including:-

4.1 The runnable agent code, packaged in a **Dockerfile** or as a Python file.

4.2 A **5-minute video** demonstrating their solution running in scenarios S6 and S7.

4.3 **2-3 pages of design notes** explaining their approach. This can be submitted in the Concept Note section. *To be submitted as part of the Product Description Document.*

4.4 The **auto-generated report** with performance metrics from the evaluator tool. *To be submitted as part of the private repository on GitHub classroom*.

NOTE-The Code may be submitted in the mentioned link-
https://classroom.github.com/a/c4_QsIKd

You need to submit your GitHub user name and link to the private repository provided by the organizer via the GitHub Classroom.