SYSC 3303 - Firefighting Drone Swarm

Lab A1 - Group 8
Abdel Qayyim Maazou Yahaya
Andrei Chirilov
Hundey Kuma
Mahad Ahmed - 101220427
Yasmin Hersi Adawe
Yusuf Ibrahim
April 8, 2025

Table of Contents

Iteration 1 Breakdown

Due Feb 1

Diagrams

Sequence: Yasmin

UML: Abdel

Members: Yasmin, Abdel, Andrei

Code

Create 3 threads:

Fire Incident Subsystem: Mahad

Drone Subsystem: Yusuf

Scheduler: Hundey

Test Files

Detail setup and instructions

Members: Mahad, Hundey, Yusuf

Iteration 2 Breakdown

Due Feb 22

Diagrams

Update Sequence: Abdel

Update UML: Yusuf

State Machine Diagram: Hundey, Abdel

3 Members: Yusuf, Hundey, Abdel

Code

Implement drone scheduling logic: Andrei

Implement drone state transitions: Yasmin

Make SRP: Mahad

Update detail setup and tests: Mahad

3 Members: Yasmin, Mahad, Andrei

Iteration 3 Breakdown

Due Mar 16

Iteration 2 Feedback:

Travel time for the drone to arrive at the zone: Hundey

Add buffer time in between events for each drone: Hundey

Calculate distance between drone and fire event (middle of the zone): Yasmin

Drone should be in idle state after refilling: Yasmin

Members: Yasmin, Hundey

Diagrams

Update Sequence: Mahad

Update UML: Andrei

State Machine Diagram: Mahad

Members: Mahad, Andrei

Code

Scheduler to coordinate movement of multiple drones with balanced workloads: Abdel

Drones to report status independently and handle commands from Scheduler: Yusuf

Implement UDP Communication for coordination between Scheduler, Fire Incident Subsystem and Drone Subsystem: Abdel

Unit tests: Yusuf, Andrei

Members: Abdel, Yusuf, Andrei

Iteration 4 Breakdown

Due Mar 29

Diagrams

Update Sequence: Yusuf

Update UML: Yusuf

State Machine Diagram: Yusuf

Timing Diagrams: Andrei

Code

Changes the event class to accept faults (change the csv and change the severity (DRONE_STUCK, NOZZLE_JAMMED, CORRUPTED_MESSAGE)) - Mahad

When FireIncident sends an incorrect event, the scheduler should send it back letting the class know that the event was incorrect and the fireIncident should send it back (add it to the back of the queue (FireIncident's queue)). - Mahad

When a drone receives a fault, depending on what type of fault it is (it should behave accordingly) - Abdel

Add timing events so that if the timer goes off before a drone reaches a zone, then your system should assume a fault - Yasmin

Unit testing - Hundey

Members: Everyone

Iteration 5 Breakdown

Due April 5

Code

Add a graphical user interface showing each drone's location, fire statuses, and any faults: Mahad, Yusuf

Record the time it takes to extinguish all fires and the distance required to reach desired zone in the input file: Yasmin

Collect and log overall performance metrics (response times, fire extinguish times, etc.): Andrei

Unit and Integration Testing: Hundey

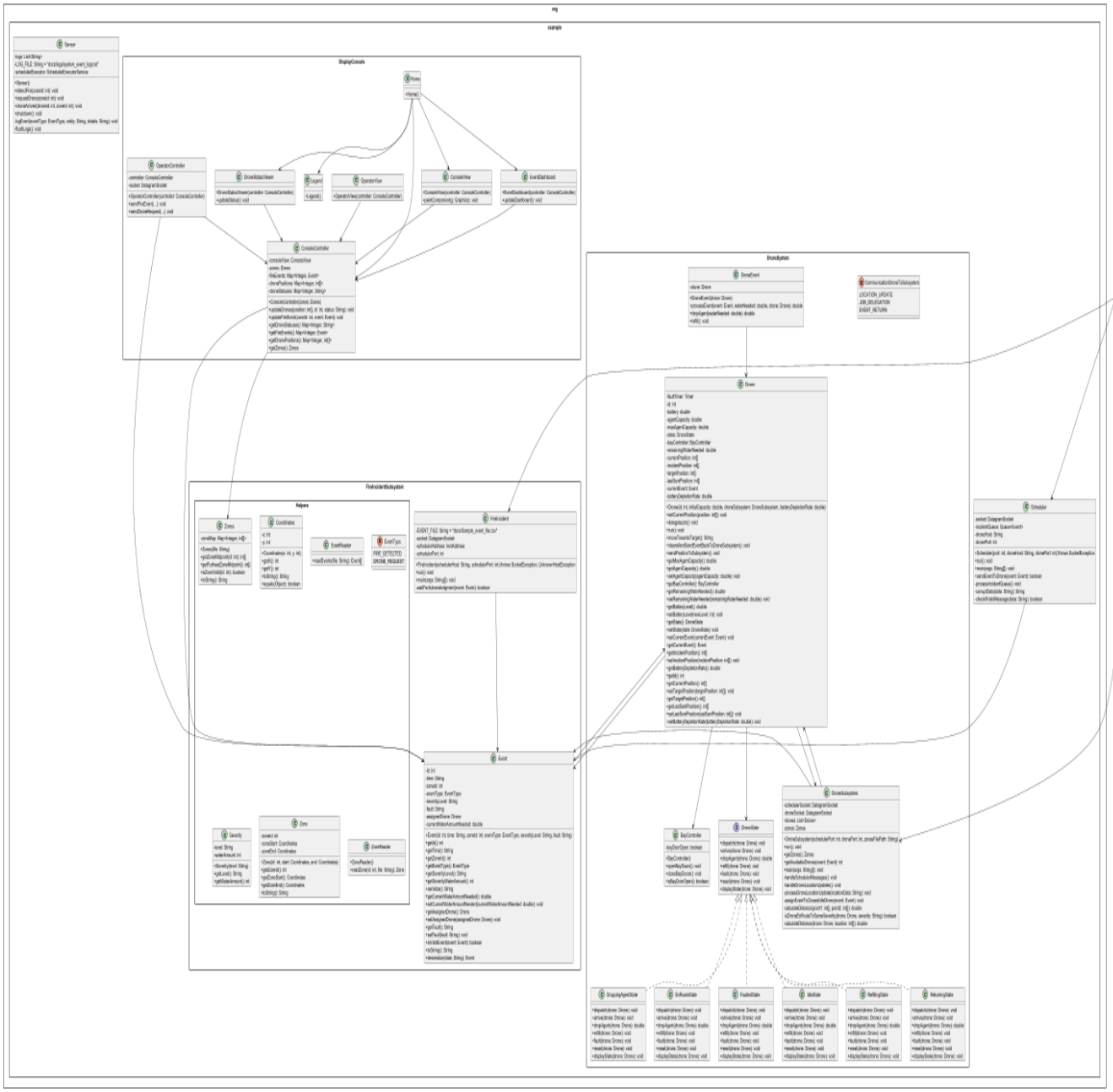
Update the drone assignment logic to ensure all 10 drones are utilized for fire zones including sending drones with remaining solution to new fire zones before returning to base, based on available battery life: Abdel

Bug testing and code cleanup: Hundey

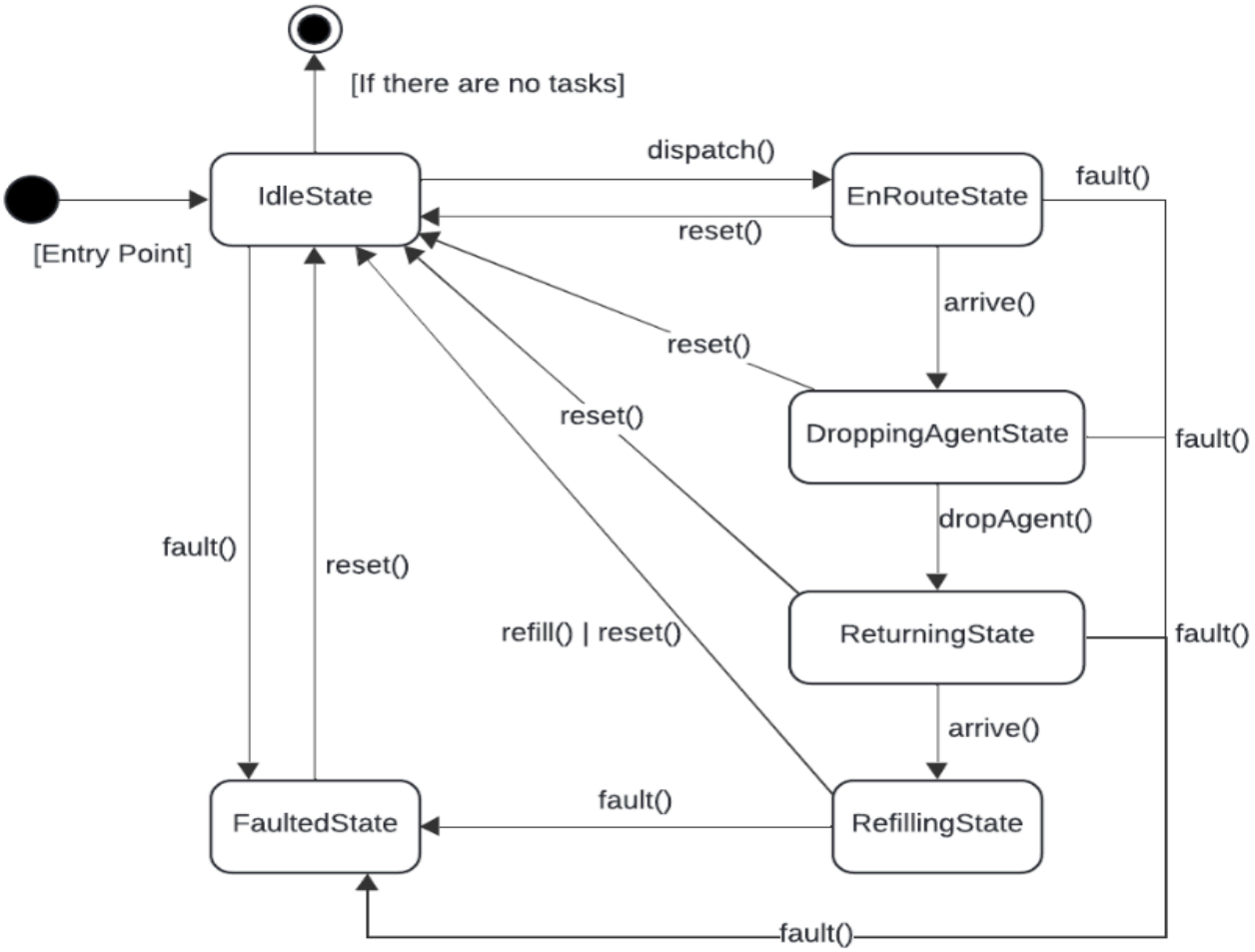
Members: Everyone

Diagrams

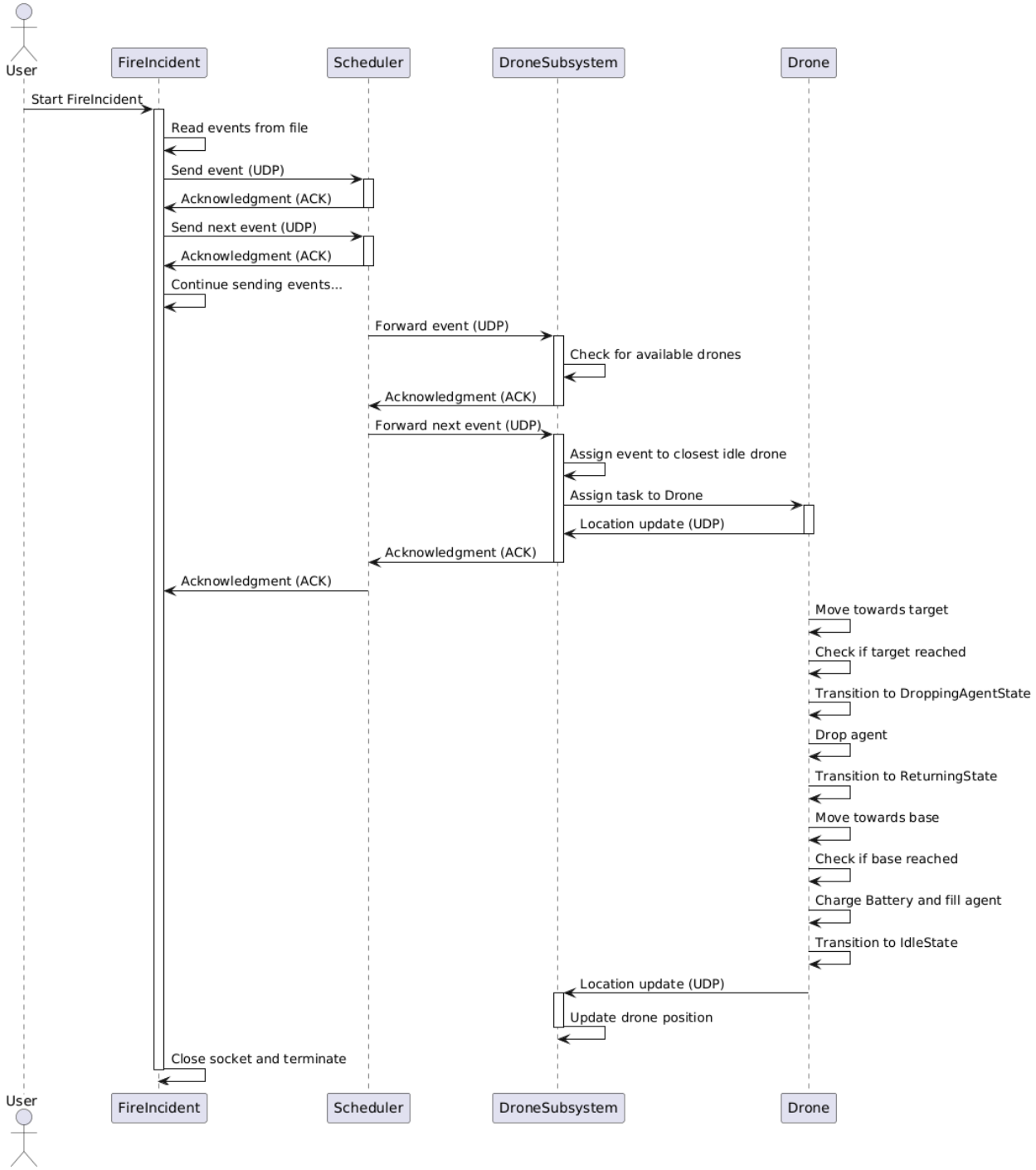
UML Class Diagram



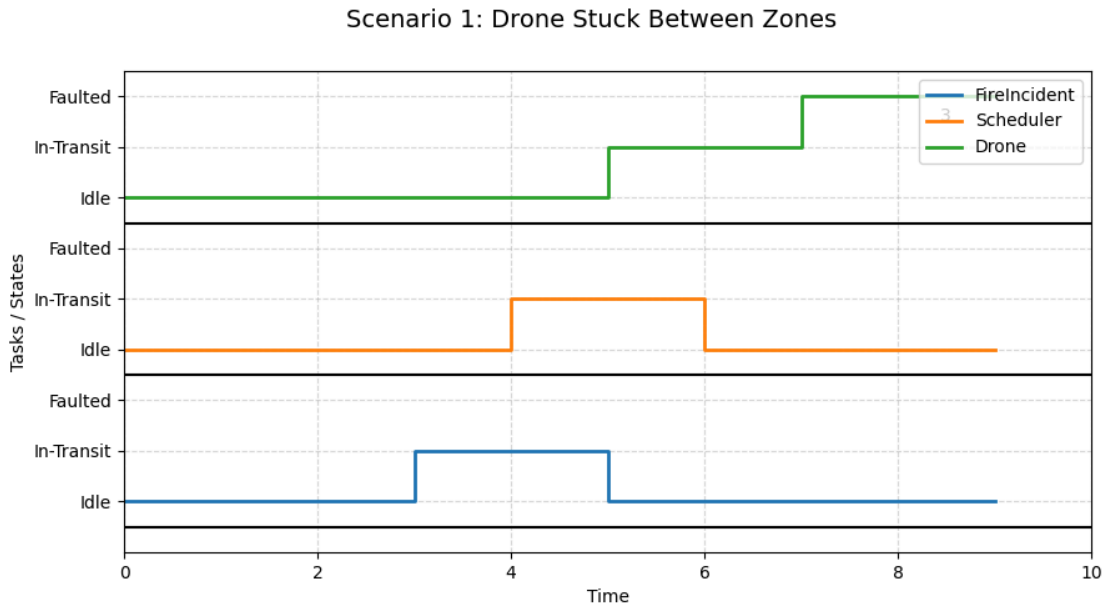
State Machine Diagram

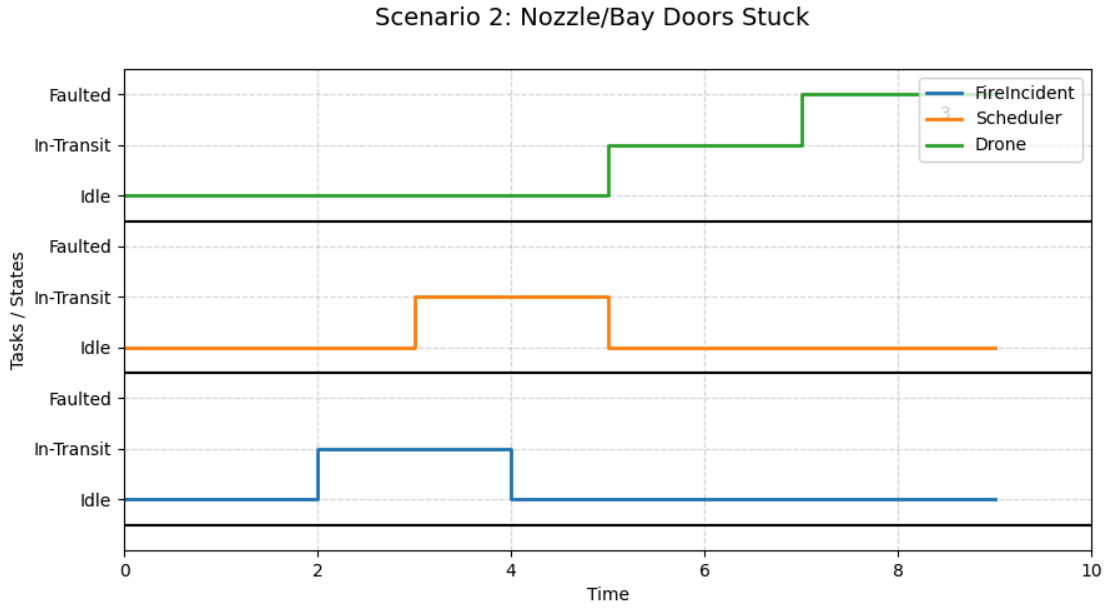


Sequence Diagram

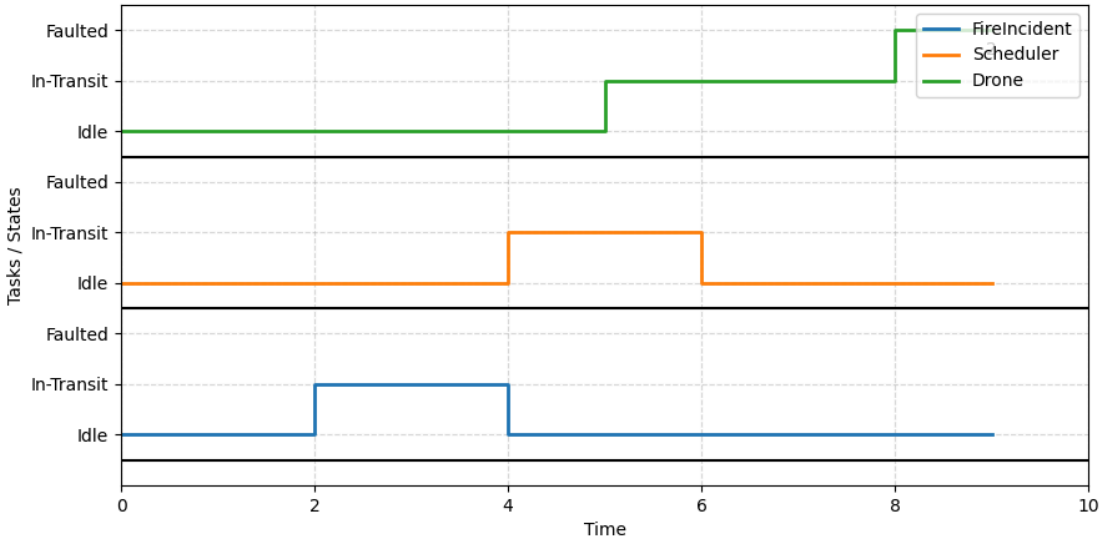


Timing Diagrams

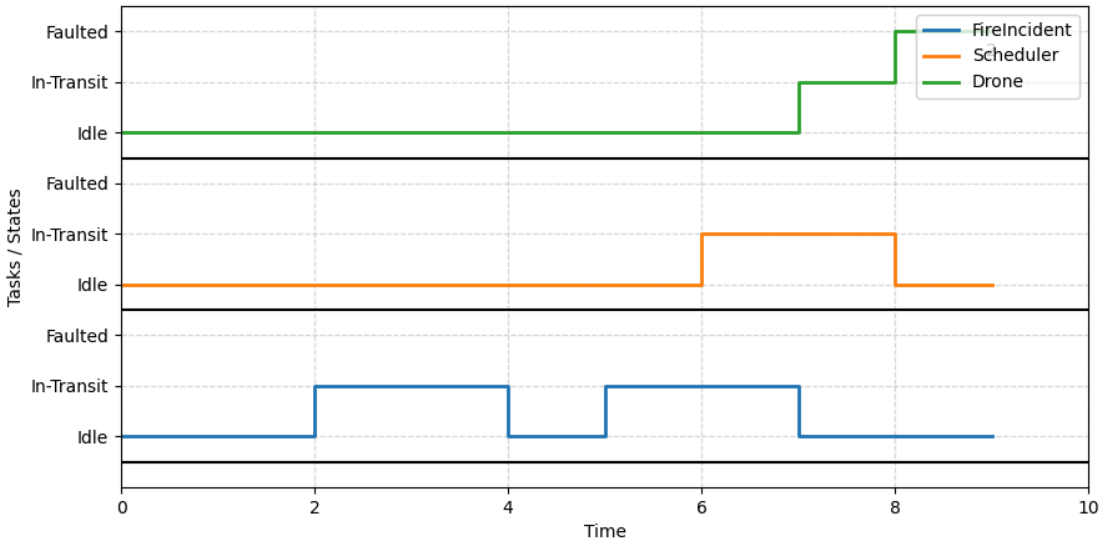




Scenario 3: Arrival Sensor Failed



Scenario 4: Packet Loss / Corrupted Messages



Guide to Setup and Tests

File Setup Responsibilities

- ➔ Main.java - Not used anymore (FireIncident, Scheduler and DroneSubsystem each have their own main class as they use UDP to communicate)
- ➔ FireIncident.java
  - ◆ Reads the incident csv file, create an event object for all the events and adds them to an events array
  - ◆ Send all the events to the schedule using UDP, everytime an event is sent, it waits for an acknowledgement from the scheduler before it sends the next one
- ➔ Fire Incident Subsystem
  - ◆ FireIncident.java - Reads fire incident events from input files and sends them to the Scheduler
  - ◆ Event.java - Represents fire/drone request event
  - ◆ EventReader.java - Helps process reading fire incident events
  - ◆ Severity.java - Helps manage Severity levels for fire incidents
  - ◆ Zone.java - Represents the zone in which the fire incident is taking place
  - ◆ ZoneReader.java - Helps process reading zone locations
  - ◆ Coordinates.java - Helps manage coordinates of fire incident zones
- ➔ Scheduler.java
  - ◆ Receives events from FireIncident class through UDP
  - ◆ Sends acknowledgement/negative acknowledgment to FireIncident if the event is valid or faulted
  - ◆ Adds the event to a queue
  - ◆ Sends the event to the DroneSubsystem and waits for response
  - ◆ If response is ACK: NO, it means there are no drones currently available
  - ◆ It will keep the event in the queue and try again every 2 seconds
  - ◆ If the event is successfully assigned to a drone, then it removes it from the queue and sends the next event in the queue
- ➔ Drone System
  - ◆ BayController.java - Controls the bay doors of the drone
  - ◆ Drone.java - Represents drone itself
  - ◆ DroneEvent.java - Handles the drone's event
  - ◆ DroneState.java - Interface represents the various states a firefighting drone can be in
  - ◆ DroneSubsystem.java - Interacts with the Scheduler to assign drones to fire incidents

- DroneSubsystem.java
  - Receives events from Schedule class through UDP
  - Attempts to assign a drone to that event
  - The criteria for assigning an event to a drone are the following
    1 - Get the closest drone to that event's location
    2 - En route drone with an event of the same severity is first
    3 - Then drones that are returning, have enough battery and agent are second
    4 - Idle drone are then next
  - If there are no drones available ACK: NO is sent back to the scheduler, meaning no drones are available for the task

- Drone.java
  - Receives an event from the DroneSubsystem
  - Travels to the event's location
  - Drops agent
  - Returns to original location to refuel (agent and batter)
  - If the fire is not put out, it will try to delegate the event to an available drone
  - Once it has refueled, it goes into the IDLE state

- Logger.java - Handles the logging and metrics functionality

- Display Console
  - ConsoleController.java - Handles logic and updates for the simulation display
  - ConsoleView.java - Draws the grid, zones, fires, and drone movements
  - DroneStatusViewer.java - Shows live drone battery and agent levels
  - EventDashboard.java - Displays active events and fire statuses
  - Home.java - Main window that combines all display components
  - Legend.java - Shows the meaning of icons and colors used in the display
  - OperatorController.java - Manages user interactions in the admin view
  - OperatorView.java - Admin panel where users can trigger and configure event

- Setup files containing the necessary data for the functionality of the simulation:
  - Sample_event_file.csv - Contains all the fire events taking place in the system
  - sample_zone_file.csv - Contains all the zones in the system
  - timings.csv - Collects the timings of the system
  - performance_report.txt - Collects the performance metrics from the logging subsystem

Test Instructions

1. Download submitted zip file
2. Extract the project and select open project with intellij (will be recognized as java project)
3. If prompted, click Import Maven Projects. If not, right-click the pom.xml file and select Add as Maven Project
4. Go to View > Tool Windows > Maven to open the Maven pane
5. In the Maven panel, click the Refresh button (a circle arrow icon) to download dependencies
6. Click on the Build menu at the top and select Build Project or press Ctrl+F9
7. Run the Scheduler class first, then the DroneSubSystem class and finally the FireIncident class
8. Running the Drone Subsystem program will open the GUI, allowing one to visualize what is going on behind the scenes with a user interface
9. Clicking on the top left of the GUI where it says “File” gives the option to generate a performance report inside “performance_report.txt”
10. There is a second window that also opens with the GUI, allowing one to add extra fire events even after the original csv input is done being all read and processed

Results from Measurements

Updated Drone Details

These are the updated values of the drone details that were collected from iteration 0:

- ➔ Travel Time Between Zones
  - ◆ Changes from 8.18s to 9.60s due to a slightly slower cruising speed (6m/s to 4m/s)
- ➔ Time to Open/Close Bay Doors
  - ◆ 3 seconds
- ➔ Rate of Dropping Water/Foam
  - ◆ Payload amount in litres * 5 seconds
- ➔ Acceleration/Deceleration for Takeoff/Landing
  - ◆ 7m/s

Performance Metrics

Using our custom csv input file for the fire incident events and zone data, this is the result of our performance metrics:

========== ALL FIRES EXTINGUISHED ==========

Event ID: 1
Response Time (ms): 4
Extinguish Time (ms): 11739
Total Time (ms): 11743
Distance Traveled: 212.13203435596427

Event ID: 2
Response Time (ms): 15699
Extinguish Time (ms): 15133
Total Time (ms): 30832
Distance Traveled: 380.7886552931954

Event ID: 3
Response Time (ms): 15783
Extinguish Time (ms): 15060
Total Time (ms): 30843
Distance Traveled: 380.7886552931954

Event ID: 4
Response Time (ms): 19069
Extinguish Time (ms): 19140
Total Time (ms): 38209
Distance Traveled: 494.9747468305833

Event ID: 5
Response Time (ms): 12011
Extinguish Time (ms): 11122
Total Time (ms): 23133
Distance Traveled: 212.13203435596427

Event ID: 6
Response Time (ms): 0
Extinguish Time (ms): 15534
Total Time (ms): 15534
Distance Traveled: 380.7886552931954

Event ID: 7
Response Time (ms): 15675
Extinguish Time (ms): 15074
Total Time (ms): 30749
Distance Traveled: 380.7886552931954

Event ID: 8
Response Time (ms): 18737
Extinguish Time (ms): 19038
Total Time (ms): 37775
Distance Traveled: 494.9747468305833

Event ID: 10
Response Time (ms): 0
Extinguish Time (ms): 16034
Total Time (ms): 16034

Distance Traveled: 380.7886552931954

Event ID: 11
Response Time (ms): 8534
Extinguish Time (ms): 9283
Total Time (ms): 17817
Distance Traveled: 380.7886552931954

Event ID: 9
Response Time (ms): 18756
Extinguish Time (ms): 36118
Total Time (ms): 54874
Distance Traveled: 494.9747468305833

Event ID: 12
Response Time (ms): 11632
Extinguish Time (ms): 11069
Total Time (ms): 22701
Distance Traveled: 212.13203435596427

Event ID: 13
Response Time (ms): 15872
Extinguish Time (ms): 16042
Total Time (ms): 31914
Distance Traveled: 380.7886552931954

Event ID: 14
Response Time (ms): 15857
Extinguish Time (ms): 15085
Total Time (ms): 30942
Distance Traveled: 380.7886552931954

Number of events: 14
Average Response Time (ms): 11973.5
Average Extinguish Time (ms): 16105.07142857143
Average Total Time (ms): 28078.571428571428
Sum of Distances: 5167.629585905205
========== END OF SIMULATION METRICS ==========

Reflection

Looking at the end product of our project, our team is particularly proud of its structure and how everything came together. Breaking the application down into its main three subsystems early on–FireIncident, Scheduler, and Drone Subsystem– allowed for clear separation of responsibilities and made debugging and feature additions in upcoming iterations more manageable. We believe we did a good job leveraging Java's object oriented features to help encapsulate functionality, as well as keeping classes focused with a clear purpose.

A specific part of the design that worked especially well was the event driven communication using UDP between subsystems. It mimics real-world distributed systems and adds realism to the simulation. Additionally, the display console added at the end was a nice touch to visualize the drone movement and event statuses in real-time, which ended up helping with some testing and debugging.

For future improvements, there are some things that can certainly be worked on. As the iterations progressed, debugging was less straightforward but this is given when developing a distributed multi-threaded system. Adding more detailed logging earlier on could have helped identify issues easier. There were also some scalability problems when having to deal with speeding up the simulation in real-time as well as properly handling all 10 drones in the 5 zones.

Overall, our team was able to overcome many problems and gained a much better understanding of concurrent systems and real-time communication. Each iteration helped teach about balancing design with implementation and documentation. Each member got the chance to experience different sides of the development of the project, whether it be working on diagrams, code or tests. The final product is a system we are proud of, and the experience gained from it will definitely be a source of knowledge for when similar situations arise.

Demo Video

Youtube link: https://youtu.be/XBQiTcKpwXo