Emergency Response: Cooperation and Coordination Mechanisms in Multi-Agent Systems

Sheena Maria Lang, Antonio Lobo Santos, Zachary Parent,
María del Carmen Ramírez Trujillo and Bruno Sánchez Gómez

December 5, 2024

Contents

1 Introduction

In this report, we present the proposed cooperation and coordination mechanisms for the CrewAI emergency response problem. The mechanisms are structured into three main components:

1. Process Definition for individual crews.
2. Pydantic Outputs for structured data handling.
3. Agent Interaction between different crews.

2 Process Definition

2.1 Emergency Services Crew

1. **Receive and Assess Call.** The *Emergency Call Agent* receives incoming calls and collects relevant details about the incident. The information that this agent receives answers the following six questions and is saved in a report:
   - What type of fire is it? E.g. ordinary, electrical, gas, etc.
   - Where is it? The location is received as coordinates (x, y).
   - Is anyone injured? How badly? The answer will be a list of strings, detailing the risk level of each person. If the list if empty then there will be no injured people and it will be unnecessary to report it to the *Medical Service Crew*.
   - How severe is the fire? It will be considered as low, medium or high.
   - Are there hazards? Examples of hazards could include gas cylinders, chemicals, explosions, etc.
   - Is it an indoor or outdoor fire? The answer will be either *outdoor* or *indoor*.
   - Is anyone inside or trapped? The answer will be an integer number M representing the number of trapped people. If $M > 0$, rescues are needed, and the *Notification Agent* will detail that to the Fire Fighters Crew.
2. **Notify Other Crews.** The *Notification Agent* receives the details about the fire then communicates the information to the appropriate crews (Medical Services Crew and Fire Fighters Crew):
   - Information provided to the Fire Fighters:
     - Location (x, y)
     - Type of fire
     - How severe is the fire
     - Number of trapped people M
     - Hazards
     - If the fire is indoors or outdoors
   - Information provided to the Medical Services Crew:
     - Location (x, y)

- Information about the injured people if any
- How severe is the fire
- Hazards

Task Dependencies: The sequential workflow for the Emergency Services Crew depends on task dependencies to ensure efficiency and coordination:

- The *Notify Other Crews Task* depends on the completion of the *Receive and Assess Call Task*.

The task dependencies and agents who perform each task can be observed in Figure 1.

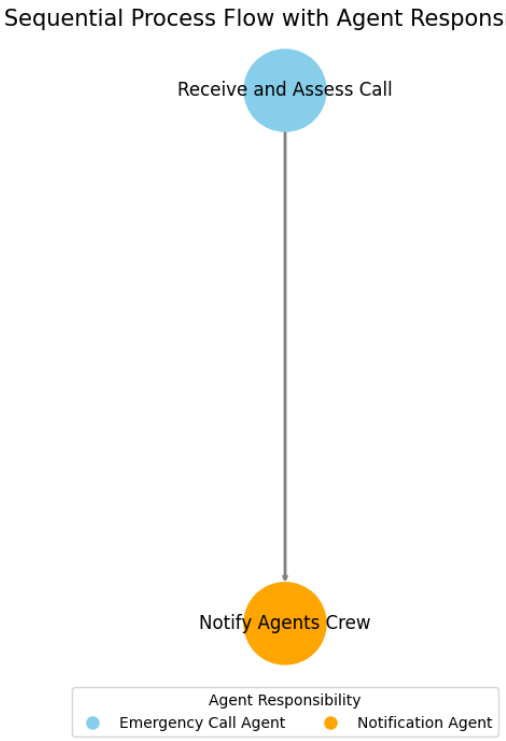

Figure 1: Sequential Process Flow of the Medical Services Crew with Agent Responsibilities

2.2 Firefighter Agent Crew

The Firefighter Agent Crew operates within a structured **sequential process** to ensure effective and coordinated response to fire emergencies. Each task is assigned to a specific agent with well-defined responsibilities, as detailed below:

1. **Receive Report:** The *Fire Chief* receives a fire report from the Emergency Service Operator. This serves as the starting point of the process, containing critical information such as the location and severity of the fire.

2. **Allocate Firefighting Resources:** The *Equipment Technician* determines if there exact resources required to combat the fire in question.

3. **Deploy Fire Combatants:** The *Fire Combatants* are deployed to the place of the fire, reporting an estimation of the time of arrival and a list of the fire fighting activities that will have to be performed.

4. **Report Firefighting Response:** The *Fire Chief* reports back a comprehensive summary of the firefighting activities as well as a success status.

Task Dependencies The sequential process relies on strict task dependencies to maintain an organized workflow:

- *Allocate Firefighting Resources* depends on the completion of *Receive Report*.
- *Deploy Fire Combatants* depends on the completion of *Deploy Fire Combatants*.
- *Report Firefighting Response* depends on the completion of *Deploy Fire Combatants*.

The visual representation in Figure 2 highlights these dependencies and assigns colors to denote the responsible agents.

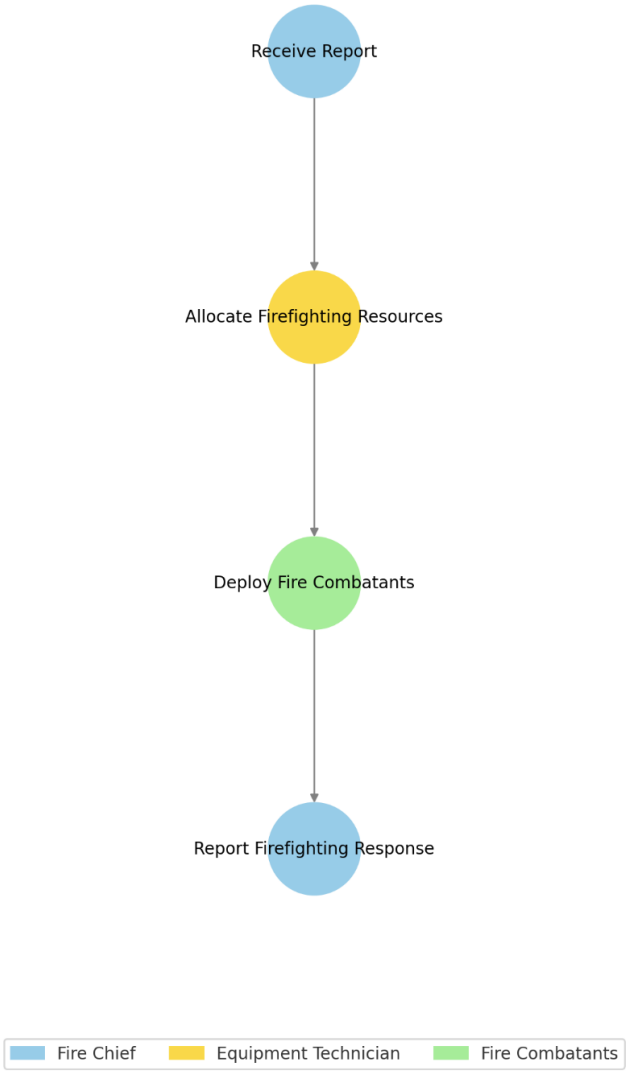

Figure 2: Sequential Process Flow of the Firefighter Crew with Agent Responsibilities

2.3 Medical Services Crew

The Medical Services Crew operates follows a **sequential** task structure to plan the treatment and evacuation of injured people from the emergency site. The tasks included within the Medical Services are:

1. **Receive Report:** The *Medical Services Operator* receives the medical report of the fire incident, and parses key information, such as the location, the number of injured, and the severity of injuries.

2. **Rank Hospitals:** The *Hospital Coordinator* ranks the city's hospitals based on distance to the emergency location.

3. **Allocate Hospital Resources:** The *Hospital Coordinator* assesses the available resources (beds, ambulances, paramedics) at the hospitals, and allocates their resources according to the needs of the emergency.

4. **Deploy Paramedics:** The *Paramedics* plan their deployment to the place of the incident, reporting the total number of paramedics and ambulances dispatched, as well as their estimated times of arrival, and any special equipment that they could need.

5. **Report Medical Response:** The *Medical Services Operator* reports back a comprehensive summary of the response plan.

Task Dependencies The sequential nature of the process requires to establish task dependencies to define the crew's workflow:

- The *Rank Hospitals* task depends on the completion of the *Recieve Report* task.
- The *Allocate Hospital Resources* task depends on the completion of *Rank Hospitals*.
- The *Deploy Paramedics* task depends on the completion of *Allocate Hospital Resources*.
- The *Report Medical Response* task depends on the completion of *Deploy Paramedics*.

The task dependencies and agents who perform each task can be observed in Figure 3.

2.4 Public Communication Crew

The Public Communication Crew operates within a structured **sequential process** to ensure efficient and accurate communication of fire incident reports to the public. Each task is assigned to a specific agent with well-defined responsibilities, as detailed below:

1. **Receive Report:** The *Communication Operator* obtains the fire incident report in Markdown format. This serves as the starting point for the process and can filter any information that is not relevant for this crew.

2. **Search Related Cases:** The *Archive Keeper* searches for past incidents with similar locations or fire types. This task depends on the completion of the *Receive Report* task.

3. **Draft Initial Article:** The *Article Writer* drafts an initial article based on the current report. This task also depends on the completion of the *Receive Report* task.

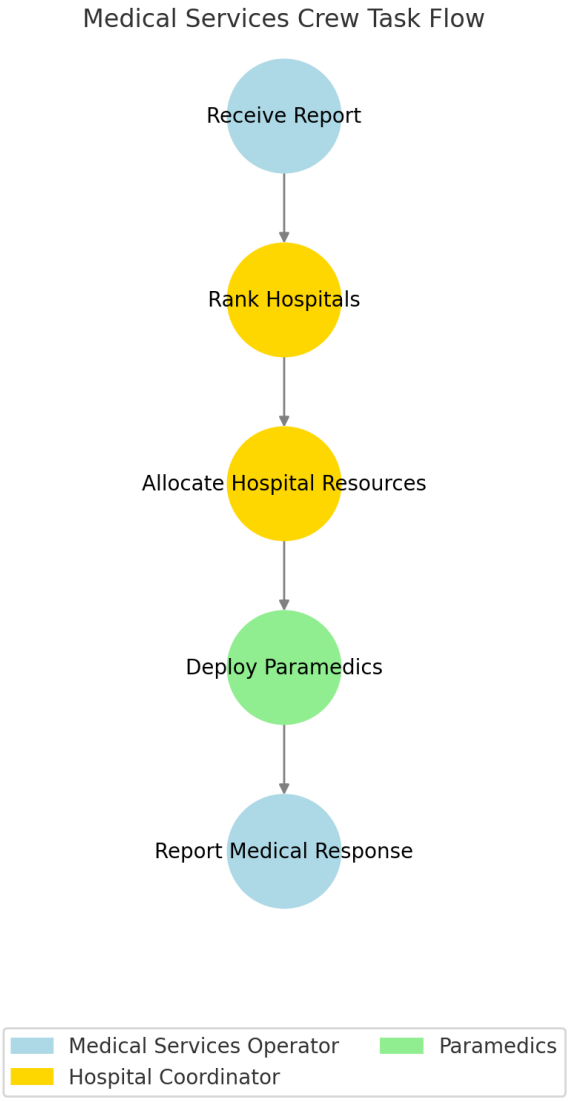

Figure 3: Sequential Process Flow of the Medical Services Crew with Agent Responsibilities

4. **Integrate Additional Information:** The *Article Writer* integrates insights from related cases into the draft. This task requires the completion of both the *Search Related Cases* and *Draft Initial Article* tasks.
5. **Review and Authorize Publication:** The *Mayor* reviews the article and either authorizes publication or provides feedback for revisions. This task depends on the completion of the *Integrate Additional Information* task.
6. **Provide Social Media Feedback:** The *Social Media Commentator* critiques the emergency response in a humorous yet constructive manner. This task depends on the approval of the article by the *Mayor*.

Task Dependencies The sequential process relies on strict task dependencies to ensure an organized workflow:

- *Search Related Cases* and *Draft Initial Article* can be executed in parallel but both depend on *Receive Report*.

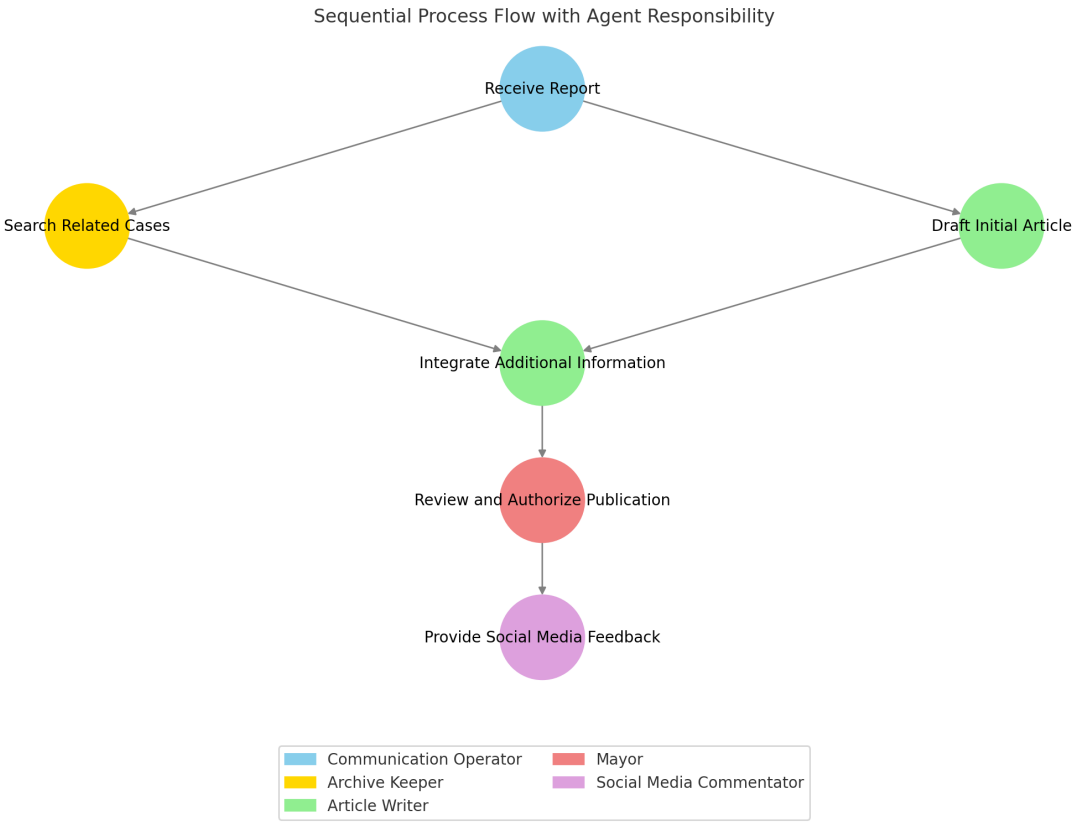

Figure 4: Sequential Process Flow of the Public Communication Crew with Agent Responsibilities

- *Integrate Additional Information* requires the completion of both *Search Related Cases* and *Draft Initial Article*.
- *Review and Authorize Publication* depends on *Integrate Additional Information*.
- *Provide Social Media Feedback* requires article approval from the *Mayor*.

The visual representation in Figure 4 highlights these dependencies and assigns colors to denote the responsible agents, ensuring clarity and accountability.

3 Pydantic Outputs

Structured outputs are essential for ensuring clarity and consistency in task execution. Below are listed the Pydantic models used in the system.

3.1 Emergency Services Crew

Structured outputs ensure accurate information handling and effective communication within the Emergency Services Crew. Below are the Pydantic models designed for each task's output.

3.1.1 Receive and Assess Call Task Output

```python
    from pydantic import BaseModel
    from typing import List, Tuple

    class CallAssessmentOutput(BaseModel):

    fire_type: str # Type of fire (e.g., ordinary, electrical, gas, etc.)
    location: Tuple[float, float] # Coordinates (x, y)
    injured_details: List[str] # List of risk levels of injured people
    fire_severity: str # Severity of fire: low, medium, or high
    hazards: List[str] # Hazards present, e.g., gas cylinders, chemicals
    indoor: bool # True if fire is indoor, False otherwise
    trapped_people: int # Number of people trapped (0 if none)
```

Listing 1: Pydantic model for Receive and Assess Call Task Output

3.1.2 Notify Other Crews Task Output

```python
    from pydantic import BaseModel
    from typing import List, Tuple

class FireReport(BaseModel):
    report_id: str
    location: Tuple[float, float] # Coordinates (x, y)
    fire_type: str # Nature of the fire (ordinary, electrical, gas, etc)
    fire_severity: str # Severity of fire: low, medium, or high
    trapped_people: int # Number of trapped individuals
    hazards: List[str] # Hazards present
    hazards_present_indoor: bool # True if fire is indoor, False otherwise

class MedicalReport(BaseModel):
    report_id: str
    location: Tuple[float, float] # Coordinates (x, y)
    injured_details: List[str] # Risk levels of injured people
    fire_severity: str # Severity of fire: low, medium, or high
    hazards: List[str] # Hazards present

class CrewNotificationOutput(BaseModel):
    firefighters_output: List[FireReport]
    medical_service_output: List[MedicalReport]
    timestamp: str # Timestamp of the notification
```

Listing 2: Pydantic model for Notify Other Crews Task Output

Summary of Outputs The Pydantic outputs for the *Emergency Services Crew* ensure structured data handling and effective communication between agents. Below is a summary

of the outputs for each task:

- **Receive and Assess Call Task Output:** Captures critical incident details including fire type, location, injured details, severity, hazards, indoor/outdoor status, and trapped individuals.
- **Notify Other Crews Task Output:** Communicates the assessed incident details to the Firefighter and Medical Services crews. Includes location, fire type, severity, hazards, and relevant medical data such as injuries.

These structured models enhance precision and ensure clear communication throughout the emergency response processes.

3.2 Firefighter Agent Crew

Structured outputs ensure effective communication and accountability among team members in the Firefighter Agent Crew. Below are the Pydantic models designed to encapsulate outputs for each task in the firefighting process:

3.2.1 Receive Report Task Output

```python
from pydantic import BaseModel

class FireReportOutput(BaseModel):
    location: Tuple[float, float] # Coordinates (x, y)
    fire_type: str # Nature of the fire
    fire_severity: str # Severity of fire: low, medium, or high
    trapped_people: int # Number of trapped individuals
    hazards: List[str] # Hazards present
    hazards_present_indoor: bool # True if fire is indoor, False
       otherwise
```

Listing 3: Pydantic model for Receive Report Task Output

3.2.2 Allocate Firefighting Resources Task Output

```python
from pydantic import BaseModel
from typing import List

class FireFightingMaterial(BaseModel):
    material_name: str
    material_quantity: int

class AllocateFirefightingResourcesOutput(BaseModel):
    resources: List[FireFightingMaterial]
```

Listing 4: Pydantic model for Allocate Firefighting Resources Task Output

3.2.3 Deploy Fire Combatants Task Output

```python
from pydantic import BaseModel

class FirefightingActivity(BaseModel):
    firefighting_activity: str
    priority_level: str

class DeployFireCombatantsOutput(BaseModel):
    firecombatants_deployed: int
    estimated_arrival_time: str
    firefighting_activities: List[FirefightingActivity]
```

Listing 5: Pydantic model for Deploy Fire Combatants Task Output

3.2.4 Report Firefighting Response Task Output

```python
from pydantic import BaseModel

class ReportFirefightingResponseOutput(BaseModel):
    summary: str
    timestamp: str
    success: bool
```

Listing 6: Pydantic model for Report Firefighting Response Task Output

Summary of Outputs

- **Receive Fire Report Task Output:** Captures the essential details from the initial fire report, including fire type, severity, hazards, and any trapped individuals.
- **Allocate Firefighting Resources Task Output:** Documents the allocation of firefighting materials, including quantities and resource types.
- **Deploy Fire Combatants Task Output:** Tracks the deployment of personnel, estimated arrival times, and prioritized firefighting activities.
- **Report Firefighting Response Task Output:** Summarizes the response outcomes, including success metrics and timestamps.

3.3 Medical Services Crew

Structured outputs ensure consistency and facilitate effective collaboration among agents within the Medical Services Crew. Below are the Pydantic models for each task's output:

3.3.1 Receive Report Task Output

```python
from pydantic import BaseModel

class MedicalReport(BaseModel):
```

```
    report_id: str
    location: Tuple[float, float]
    injured_details: List[str]
    fire_severity: str
    hazards: List[str]

class ReceiveReportOutput(BaseModel):
    report: MedicalReport
    timestamp: str
```

Listing 7: Pydantic model for Receive Report Task Output

3.3.2 Rank Hospitals Task Output

```
from pydantic import BaseModel

class Hospital(BaseModel):
    hospital_id: str
    location: str
    available_beds: int
    available_ambulances: int
    available_paramedics: int

class RankHospitalsOutput(BaseModel):
    report: MedicalReport
    ranked_hospitals: List[Hospital]
    timestamp: str
```

Listing 8: Pydantic model for Rank Hospitals Task Output

3.3.3 Allocate Hospital Resources Task Output

```
from pydantic import BaseModel

class HospitalResources(BaseModel):
    hospital_id: str
    beds_reserved: int
    ambulances_dispatched: int
    paramedics_deployed: int

class AllocateHospitalResourcesOutput(BaseModel):
    report: MedicalReport
    hospital_resource_allocation = List[HospitalResources]
    timestamp: str
```

Listing 9: Pydantic model for Allocate Hospital Resources Task Output

3.3.4 Deploy Paramedics Task Output

```python
from pydantic import BaseModel
from typing import List

class MedicalEquipment(BaseModel):
    equipment_name: str
    use_case: str

class DeployParamedicsOutput(BaseModel):
    report: MedicalReport
    total_paramedics_deployed: int
    total_ambulances_dispatched: int
    estimated_arrival_times: List[str]
    equipment: List[MedicalEquipment]
```

Listing 10: Pydantic model for Deploy Paramedics Task Output

3.3.5 Report Medical Response Task Output

```python
from pydantic import BaseModel

class MedicalResponseReportOutput(BaseModel):
    summary: str
    timestamp: str
```

Listing 11: Pydantic model for Report Medical Response Task Output

Summary of Outputs

- **Receive Report Task Output:** Captures the key details of the fire incident, including injury data.
- **Rank Hospitals Task Output:** Ranks the available hospital based on distance to the emergency site.
- **Allocate Hospital Resources Task Output:** Summarizes the resources provided by each hospital for emergency medical care.
- **Deploy Paramedics Task Output:** Reports the deployment plan, estimated times of arrival of each ambulance, and special medical equipment to be brought.
- **Report Medical Response Task Output:** Provides an overall response plan.

3.4 Public Communication Crew

Structured outputs are crucial for ensuring clarity, consistency, and seamless integration across tasks. Below are the Pydantic models designed for the tasks in the Public Communication Crew process:

3.4.1 Receive Report Task Output

```python
class ReceiveReportOutput(BaseModel):
    location: Tuple[float,float]
    fire_severity: str
    timestamp: str
    markdown_content: str
```

Listing 12: Pydantic model for Receive Report Task Output

3.4.2 Search Related Cases Task Output

```python
class RelatedCase(BaseModel):
    case_id: str
    location: Tuple[float,float]
    fire_severity: str
    summary: str

class SearchRelatedCasesOutput(BaseModel):
    related_cases: List[RelatedCase]
    total_cases: int
```

Listing 13: Pydantic model for Search Related Cases Task Output

3.4.3 Draft Initial Article Task Output

```python
class DraftArticleOutput(BaseModel):
    title: str
    draft: str
```

Listing 14: Pydantic model for Draft Initial Article Task Output

3.4.4 Integrate Additional Information Task Output

```python
class IntegratedArticleOutput(BaseModel):
    draft: str
    integrated_sources: list[str]
```

Listing 15: Pydantic model for Integrate Additional Information Task Output

3.4.5 Review and Authorize Publication Task Output

```python
class ReviewOutput(BaseModel):
    approved: bool
    comments: str
    report: str
```

Listing 16: Pydantic model for Review and Authorize Publication Task Output

3.4.6 Provide Social Media Feedback Task Output

```python
class SocialMediaFeedbackOutput(BaseModel):
    feedback: str
    report: str
    approved: bool
    comments: str
```

Listing 17: Pydantic model for Provide Social Media Feedback Task Output

Summary of Outputs

- **Receive Report Task Output:** Captures the initial fire incident report relevant details from *Firefighters* and *Medical Sevices* crews.
- **Search Related Cases Task Output:** Retrieves relevant historical cases for contextualization and save this case.
- **Draft Initial Article Task Output:** Records the initial draft content.
- **Integrate Additional Information Task Output:** Updates the draft with integrated sources and revisions.
- **Review and Authorize Publication Task Output:** Specifies the review status and comments from the Mayor.
- **Provide Social Media Feedback Task Output:** Details feedback posted on social media platforms, he can critize the mayor's decission.

4 Agent Interaction

4.1 Interaction Flow

Interaction between crews is designed using a flow-based approach:

- The Search and Rescue Crew provides victim data to the Medical Response Crew.
- A **Router** determines the priority of medical cases based on data received.

4.2 Router Implementation

The router ensures tasks are efficiently allocated:

```python
def router(victim_data):
    if victim_data["priority"] == 1:
        return "Critical Response Team"
    else:
        return "General Response Team"
```

Listing 18: Router Implementation for Task Allocation

4.3 Communication Mechanism

Agents communicate using a message-passing protocol to ensure scalability. An example interaction is illustrated in Figure **??**.

5 Conclusion

This report outlines the proposed cooperation and coordination mechanisms for the CrewAI MAS. Future work includes extending the interaction model to include additional agent types and testing the scalability of the system.