A Decentralized Agent-based Model for Crisis Events Using Embedded Systems

Nilson Mori Lazarin[1,2], Tielle Alexandre[1], Magaywer Moreira de Paiva[1], Carlos Eduardo Pantoja[2], Jose Viterbo[1], and Flavia Bernardini[1]

[1] Institute of Computing - Fluminense Federal University (UFF), Niterói - RJ, Brazil
{magaywermp,tiellesa}@id.uff.br {viterbo,fcbernardini}@ic.uff.br
[2] Federal Center for Technological Education Celso Suckow da Fonseca (Cefet/RJ), Rio de Janeiro - RJ, Brazil
{nilson.lazarin,carlos.pantoja}@cefet-rj.br

Abstract. Multi-agent systems (MAS) applied to Embedded Systems enable cognitive agents to act in the physical world. However, the application of these systems has been little explored to automate communication during crisis events. With this approach, it would be possible to help collect real-time data and deploy rescue forces in risky locations. This paper describes a decentralized, proactive, and agent-based communication model. To validate the proposal, we applied our approach to a physical prototype of smart city devices. In this scenario, we include the interaction of different MAS in a simulation's flood scenario. The results show the architecture's effectiveness in collecting real-time data, providing a reliable, low-cost way to integrate citizens and governments during crises. The architecture is a promising approach for crisis management applications that takes advantage of the autonomous behavior of agents in MAS, improving the obtaining of subsidies to feed systems in smart cities and assisting decision-making.

Keywords: Multi-agent systems · Crisis Events · Embedded Systems.

1 INTRODUCTION

The advancement of communication technology and computing devices has driven the development of Smart Cities during the past decade. Its main focus is to improve citizens' quality of life, and this can be done by implementing solutions for monitoring and controlling daily problems faced by people in cities. The United Nations proposed Sustainable Development Goals, which Goal 11, in particular, aims to make cities and human settlements more resilient, safe, and sustainable. In this sense, ensuring a quick and coordinated response to some critical problems cities face is crucial for preserving the safety and well-being of citizens.

Crisis events can disrupt city environments, affecting safety, resilience, and sustainability. Among these events, floods are the most widespread natural disaster globally [24]. The negative effects caused by floods can be observed in developed and least developed countries [1], posing substantial risks to human

life, infrastructure, agriculture [7,25], environmental sustainability [13], and socioeconomic systems [14]. Floods are characterized by water accumulation in typically dry points or regions or overflowing typically flooded areas [6].

Cities can implement weather forecasting systems, rain gauges, water level sensors, and other technologies to provide data to facilitate a coordinated and effective response to flood events and overcome these challenges. Supported by available data, different distributed systems can communicate in real-time, sharing information to monitor climatic and hydrological conditions. Efficient communication between different systems can favor the construction of safer and more resilient environments, especially during events' crises. However, the complexity of urban scenarios and the vast number of interconnected devices require an architecture that guarantees integrated and intelligent solutions. Sometimes, distributed and edge solutions can be more effective since the first decision-making can happen in the bare local instead of waiting for authorities.

The Multi-Agent Systems (MAS) are composed of autonomous agents capable of perceiving and acting in their environment, reasoning based on cognitive models, and communicating with other agents [26]. These characteristics make agents adaptable and capable of dealing with dynamic scenarios [10]. Agents have been applied in many areas, from traffic to resource management (e.g., energy and water). Besides, MAS has been used in embedded systems and IoT scenarios to control the sensors and actuators of a device. They can collect and analyze real-time data, identify patterns and anomalies, and act collaboratively to optimize a collective goal by interacting with other distributed and embedded MAS [4]. In this sense, using agent-based embedded systems presents a promising approach to addressing smart cities' challenges.

This paper presents the Decentralized Agent-based Response Crisis Model (DARC), an autonomous and scalable approach to dealing with generic crisis events, supporting the response coordination actions integrating MAS from government agencies and Embedded MAS of smart homes. To this, JasonEmbedded [16], a spin-off of Jason [3] that integrates the Internet of Things and Embedded Systems was adopted. It allows the scalability of devices, bandwidth-saving interactions, and asynchronous communication to provide embedded agents with social interactions over large-scale systems. Finally, we present a case study using the ALERT-AS from the National Institute of Meteorology (INMET) — a Brazilian agency, two MAS representing governance instances, and an embedded MAS representing a smart home. The main contributions of this work are a model for crisis events using distributed Embedded MAS and an example using real meteorological data. Therefore, this work is organized as follows: Section 2 presents the related works; Section 3 presents the proposed model; Section 4 presents a case study; finally, the considerations are presented in Section 5.

2 Related Works

A review [22] of the use of MAS to solve flooding problems, their management, evaluation, and efforts to predict stream flow and flood events has shown that

MAS presents benefits and advantages in distributed artificial intelligence and in dealing with rule-based resources. Another study [12] suggests that in the case of a flood disaster where long-term flooding is expected, it is effective to use agent systems to respond to a dynamic environment because they can adjust transportation methods and tactics in response to damage and disaster scenarios. Besides, MAS was used to analyze the evolution of water-sediment connectivity in a small agricultural area to address the problem of flash flooding, providing a better understanding of the challenges related to flash flooding and helping to develop more effective management strategies [19].

Yaning and Qianwen [27] presented an Internet-driven data management system, working as crowdsourcing for crisis management, to improve the information sharing of resources in collaborative co-governance of public crisis management. They showed a MAS-based innovation to achieve a higher level of joint prevention and control mechanisms for public crisis management. Sasaki and Kitsuya [20] proposed a Regional Information Sharing System to reduce disaster risk. A simulation of the behavior of evacuees during a disaster in a specific area, using MAS, including three types of agents: general evacuees, vulnerable people, and supporters, showed that the proposal can help reduce the number of casualties caused by non-existent or late evacuation and support. Marouane [11] implemented a distributed decision support system to help decision-makers manage crises. It used satellite images, remote sensors, geographic information systems, and databases integrated with a MAS to constrain the smartest decisions possible. Finally, Rafanelli et al. [18] showed a MAS specialized in monitoring flood events, combined with a neural network that inspects and analyzes satellite images. The agents generate alerts in the early detection of flood events, seeking to help mitigate flood damage.

The use of MAS has proven to be comprehensive in this topic; however, no studies using Embedded MAS applied to crisis management have been found in some reviews [22,12,19]. We argue that the use of the embedded system can bring benefits. We argue that the use of embedded systems can bring benefits. For example, different from Yaning and Qianwen [27], we present agent-based crowdsourcing since a smart home using embedded MAS can contribute to public crisis management by sending environmental perceptions automatically and in real-time; different from Sasaki and Kitsuya [20], each embedded MAS can receive instructions from authorities in real-time, transforming them into actions in the physical world, such as alerts directly to citizens' homes; finally, this paper proposes a low-cost and an agent-based decentralized model, that does not need to use satellite images and geographic information systems, different from Marouane and Rafanelli et al. [11].

3 Proposed Model

An open MAS allows agents to enter or leave the agent society at any moment without any global control. It allows the formation of Coalitions, a dynamic organizational structure where the agents may collaborate to achieve their goals,

creating a coherent behavior to solve complex tasks [21]. This paper proposes the Decentralized Agent-Based Response Crisis (DARC) Model, based on embedded MAS, open MAS, and coalitions that provide an autonomous approach to deal with generic crisis events and support the coordination of response actions by integrating government agencies and smart homes. With this model, when disaster events occur (or are expected), such as flooding, landslides, windstorms, fire, etc., the accountable government agency can propose a multi-agent coalition formation (like a task force) to manage each event specifically. In addition, cognitive agents from other involved government agencies can move to the coalition to help organize the necessary actions.

This model aims to improve the decision-making because a coalition can be formed for micro-regions specifically, and scalability because each coalition can be hosted in a different computer infrastructure (e.g., IoT device). A Low-End IoT device is a system constrained in computational resources, manufactured for basic sensing and actuating applications, and programmed using low-level firmware to few functionalities [15]. These devices cannot host a MAS but can be managed by cognitive agents using specific serial communication protocols [9]. Conversely, a High-End IoT device is a single-board computer with enough computational resources to run a traditional operating system [15]. It allows the embedding of systems using an agent-based approach [16].

The mobile agents from coalitions carry their specific knowledge (beliefs, intentions, plans, and goals), acting like a bridge between their government agency and the other agencies working together in the coalition to solve a specific crisis event. Furthermore, smart homes obtain information from public alert services. When an event is expected for a location, following the alert guidelines, the smart home consults the respective government authority about the event, asking which coalition is responsible for coordinating that event. It also updates the coalition in real-time with information that can assist in decision-making, such as occupants, people with special needs, or even sensing information in loco that could help accurately define the affected areas. Additionally, the coalition can provide alert and guidance on possible evacuation needs specific to each micro-region. Figure 1 shows the proposed model, defined as $DARC = \{\overrightarrow{C}, \overrightarrow{F}, \overrightarrow{G}, \overrightarrow{H}\}$, where:

- J is a *common* autonomous cognitive agent;
- A is a low-end IoT device that acts in the environment;
- S is a low-end IoT device that perceives the environment;
- J_A is an *actuator* agent, a cognitive agent capable of sensing and acting in the exogenous environment (physical world);
- J_C is a *communicator* agent, a cognitive agent capable of exchanging messages and migrating agents between open MAS;
- F represents any crisis event forecasting service offered digitally, which provides information about the environment (e.g., places at risk for heavy rain, landslides, or river flooding);
- J_F is a *forecaster* agent, a cognitive agent capable of searching for alerts in the forecast services;

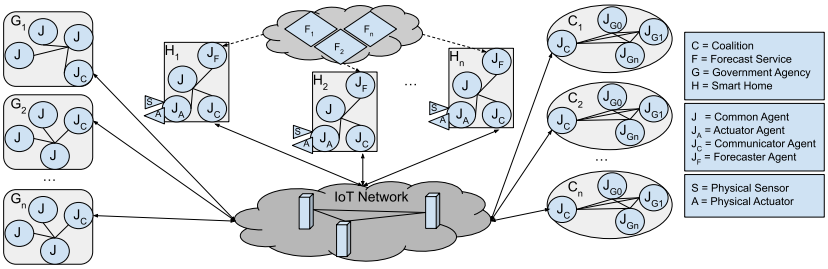

Fig. 1. Decentralized Agent-Based Response Crisis (DARC) Model.

- $G_j = \{J_C, J_1, J_2 \cdots J_n\}$ is a MAS hosted by a government agency (e.g., civil defense, firefighters);
- $H_k = \{J_C, J, J_F, (J_{A0}, J_{A1} \cdots J_{An}) \mid \forall J_A \exists (A \vee S)\}$ is an embedded MAS that manages a smart home;
- $C_i = \{J_C, (J_{G1}, J_{G2}, \cdots, J_{Gn}) \mid J_{G1} \in G_1, J_{G2} \in G_2, \ldots, J_{Gn} \in G_n\}$ is an open MAS formed by a coalition of agents from different government agencies (J_G). These coalitions deal with a specific event, supporting public crisis management.

4 Case Study

To evaluate the model, we propose a case study that considers a city close to a river where the population is always subject to flooding during the heavy rainy season. In compliance with UN Objective 11, to face these situations, government agencies used MAS to automate communication and monitoring in crisis events and assist in the decision-making process. The smart homes around the river are equipped with embedded MAS with sensors that check the intensity of rain and actuators that issue a visual alert about a possible flood.

Furthermore, they have agents capable of searching for information about their region through a national weather alert service. This continuous interaction between the different MAS allows the city to prepare adequately to face flooding risks, protecting the population and minimizing damage caused by floods. This scenario was planned to create a practical demonstration of how Embedded MAS technology can be applied to improve the safety and resilience of a community in the face of emergencies.

To demonstrate the model in practice, we present a scenario where a publicly known MAS hosted by the Civil Defense is initially accountable for supporting all running coalitions and handling each alert issued by the meteorological service. The City Hall, in your turn, hosts a MAS responsible for registering the city with the national civil defense MAS and awaiting possible demands arising from meteorological alerts. Once the city hall's MAS is notified about a crisis event, a coalition is formed to deal with the specific event. The coalition comprises a

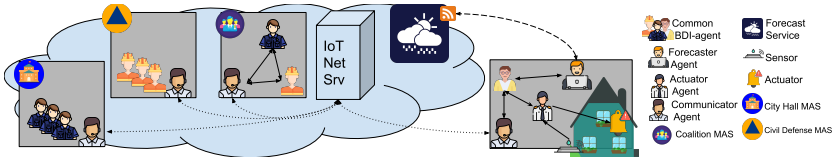

Fig. 2. Flood alert and communication in a Smart City scenario.

coordinating agent sent by the city hall's MAS and an assistant agent sent by the civil defense MAS.

An Embedded MAS is hosted in the smart home. When perceiving that an alert has been issued for their location, they contact the civil defense MAS, requesting information about which coalition is handling that alert. With this information, the smart home informs the coalition and waits for instructions. In the newly formed coalition, the assistant agent is accountable for sending the necessary guidance to the smart home. These, in turn, start to inform what was requested by the assistant agent in real-time.

Suppose the assistant understands that the information provided by the smart home justifies a change to the alert status. In that case, the assistant will request that the actuator agent in the smart home issue an alert to the residents. In an emergency requiring evacuation, the assistant agent will inform the coordinating agent that a government agency intervention will be necessary for that specific smart home. The coordinating agent, in turn, has information about the staff currently active and the city hall's response capabilities to manage demands and will forward rescue requests to the city hall's MAS.

4.1 Implementation

To implement the scenario, we used the JasonEmbedded [16] framework, a spin-off version of Jason [3] that integrates the Internet of Things, Embedded Systems, and Distributed Artificial Intelligence paradigms. It allows different architectures of agents in the development process of the MAS, such as: i) Jason [2] is an agent based on the Belief-Desire-Intention (BDI) [5] model that fulfills the expected autonomous cognitive agent J; ii) ARGO [17] is an agent capable of perceiving and acting directly in the exogenous environment, using Javino [9] serial communication protocol, fulfilling the expected actuator agent J_A; iii) Communicator [23] is an agent capable of exchanging messages and migrating agents between different MAS, hosted on different computers, geographically distributed, using the ContextNet [8] IoT middleware that fulfills the expected by agent J_C.

However, as the framework does not support creating coalitions, nor does it have an expected forecaster agent J_F. First, it was necessary to extend the framework to fulfill the proposed scenario. After that, three MAS were implemented: one representing the civil defense agency, one representing the city hall, and finally, an Embedded MAS representing the smart home, described below:

• ***Civil Defense's MAS:*** The MAS hosted by Civil Defense has a *Communicator* agent (named *telephonist* — presented in the Code 1.6) and a *Jason* agent (named *commander*), who is presented in the Code 1.1. The agent *commander* has three plans:

- The first plan (*registerCity*) creates a mental notation with the identification code and IoT address of the MAS hosted at the city hall;
- The second plan (*getInfo*) has two possible contexts: in the case where the smart home seeks information about an alert that has not yet been notified, then the agent requests to agent *telephonist* (*.send* internal action) forwarding a message to the *mayor* agent in the city hall's MAS to proceed with the coalition formation to address the reported alert; and if the coalition already exists, then the agent requests the sending of a message to the *leader* agent in the smart home's MAS with the coalition's address in the IoT network;
- The last plan (*taskForce*) is triggered when the agent receives the belief that a new coalition has been successfully formed, so it instantiates a new agent in the MAS (*.create_agent* internal action) with the assistant's role and requests to *telephonist* the transport of this agent for the newly created coalition MAS.

```
1  +!registerCity(IBGE,CityIotAddress)[source(telephonist)] <-
       +cityDirectory(IBGE,CityIotAddress);
3      .send(telephonist,achieve,sendExternalMessage(CityIotAddress,mayor,tell,registredInCivilDefense)).
   +!getInfo(Alert,IBGE,House)[source(telephonist)]: taskForce(TFnr,City,TFuuid) & Alert = TFnr <-
5      -preparingTaskForce(TFnr);
       .send(telephonist,achieve,sendExternalMessage(House,leader,tell,taskForce(TFnr,TFuuid))).
7  +!getInfo(TFnr,IBGE,_)[source(telephonist)]:not preparingTaskForce(Alert)& not taskForce(TFnr,City,TFuuid)<-
       ?cityDirectory(IBGE,CityIotAddress);
9      .send(telephonist,achieve,sendExternalMessage(CityIotAddress,mayor,achieve,createTaskForce(TFnr))).
   +taskForce(TFnr,City,TFuuid) <-
11     .create_agent(assistant,"agents/roles/eventAssistant.asl");
       .send(telephonist,achieve,tranportAnAgentTo(assistant,TFuuid)).
```

Code 1.1. Implementation of agent *commander* in civilDefense MAS.

• ***City Hall's MAS:*** The MAS hosted by City Hall has a *Communicator* agent (named *telephonist* — presented in the Code 1.6) and a *Jason* agent (named *mayor*), who is presented in Code 1.2. The agent *mayor* has only one goal and five plans:

- The initial goal triggers the first plan (*registerInCivilDefense*) that tries to register the city and its IoT address with the agent *commander* in the civil defense's MAS.
- The second plan (*createTaskForce*) is triggered when the agent receives a message from the *commander* agent in civil defense' MAS about the occurrence of an alert. In this case, the agent *mayor* must create a new MAS to host the coalition (*.create_mas* — internal action proposed in this paper). While the new MAS is in the orchestration process, the agent *mayor* creates a new agent (*coordinator*) to represent the city hall in the coalition (*.create_agent* internal action). After that, the agent *mayor* transmits to the new agent all the necessary knowledge to deal with the crisis event (lines 9-14).
- The third plan (*newMasSuccessfullyCreated*) is triggered when the initial agent of the newly created MAS can connect with the IoT gateway and inform the *telephonist* that the new MAS was instantiated correctly.

– The fourth plan (*alert*) shows a message that the city has been notified about the need for a rescue in a certain smart home.
– Finally, the last plan (*sendCoordinatorToMAS*), requests to *telephonist* the transfer of the agent *coordinator* to the newly created MAS and the immediate activation of the coalition once the open MAS is already.

```
!registerInCivilDefense.    /* initial goal */
/* plans */
+!registerInCivilDefense: nationalCivilDefense(CDuuid) & myIotAddress(CITYuuid) & myCityIBGE(IBGE) <-
    .send(telephonist,achieve,sendExternalMessage(CDuuid,commander,achieve,registerCity(IBGE,CITYuuid))).
+!createTaskForce(Alert): not waitCallBack(Alert) & nationalCivilDefense(CDuuid) & myIotAddress(CITYuuid)<-
    .send(telephonist,achieve,sendExternalMessage(CDuuid,commander,tell,preparingTaskForce(Alert)));
    .create_mas("extra/newCoalition.zip",gui,callBack(CITYuuid));   +waitCallBack(Alert);
    .create_agent(coordinator,"../_commonAgents/citizen.asl");
    .send(coordinator,tell,nationalCivilDefense(CDuuid));
    .send(coordinator,tell,alertID(Alert));
    .send(coordinator,tell,cityHallIotAddress(CITYuuid));
    .send(coordinator,tellHow,"+!startTaskForce <- .send(telephonist,askOne,myIotAddress(U),Reply); +Reply;
      !registerTaskForce.");
    .send(coordinator,tellHow,"+!registerTaskForce: myCityIBGE(IBGE)& alertID(ID) & nationalCivilDefense(CD)
       & myIotAddress(U) <- .send(telephonist,achieve,sendExternalMessage(CD,commander,tell,taskForce(ID,
      IBGE,U))).");
    .send(coordinator,tellHow,"+evacuating(Home): smartHome(UUID) & UUID=Home & alertID(ID) &
      cityHallIotAddress(CityUUID) <- .send(telephonist,achieve,sendExternalMessage(CityUUID,mayor,tell,
      alert(evacuate,Home))).");
    !sendCoordinatorToMAS(Alert).
+!newMasSuccessfullyCreated(UUID)[source(telephonist)]: waitCallBack(Alert)[source(self)]<-
    .send(telephonist,achieve,mas2masMessage(UUID,tell,where(Alert)));
    -waitCallBack(Alert);
    +taskForce(Alert,UUID).
+alert(evacuate,Home) <-  .print("Preparing Rescue for ",Home).
+!sendCoordinatorToMAS(Alert): taskForce(Alert,UUID) <-
    .send(telephonist,achieve,tranportAnAgentTo(coordinator,UUID)); .wait(10000);
    .send(telephonist,achieve,sendExternalMessage(UUID,coordinator,achieve,startTaskForce)).
```

Code 1.2. Implementation of agent *mayor* in cityHall MAS.

• ***Smart Home's MAS:*** The MAS hosted by Smart Home has a *Communicator* agent (named *telephonist* — presented in the Code 1.6), two *Jason* agents presented in Code 1.3 (named *leader*) and Code 1.4 (named *forecaster*), finally has an *ARGO* agent (named *actuator*) who is presented in Code 1.5. The agent *leader* has four plans:

– The first (*weatherEvent*) is activated when the agent *forecaster* sends a belief representing that an event is predicted for the micro-region.
– The second plan (*getCivilDefenseInfo*) requests to *telephonist* ask for information about the event to civil defense' MAS.
– The third plan (*taskForce*) is activated when civil defense' MAS reports that there is a coalition dealing with the alert. In this case, the agent will inform all agents in the smart home (*.broadcast* internal action) that they are waiting for instructions; requests to *actuator* to turn on a LED indicating that the embedded MAS is online; and finally, requests to *telephonist* to forward the house information to the coalition.
– The last plan (*alert*) requests to *actuator* to turn on the LED indicating the alert. There are two contexts: the first is a case of attention, and the second is a case of imminent flooding.

```
+weatherEvent(Alert,Event,Serverity)[source(forecaster)]: not contatingCivilDefense<-
    +contatingCivilDefense; !getCivilDefenseInfo(Alert).
+!getCivilDefenseInfo(Alert): nationalCivilDefense(CDuuid) & myCityIBGE(IBGE) & myIotAddress(Uuid) <-
    .send(telephonist,achieve,sendExternalMessage(CDuuid,commander,achieve,getInfo(Alert,IBGE,Uuid))).
+taskForce(Alert,TaskForceUUID): myCityIBGE(IBGE) & myIotAddress(Uuid) <-
    .broadcast(tell,waitingInstructions(Alert,TaskForceUUID,Uuid));
```

```
7       .send(actuator,achieve,greenAlert);
        .send(telephonist,achieve,sendExternalMessage(TaskForceUUID,coordinator,tell,smartHome(Uuid))).
9   +!alert(ID,evacuate): taskForce(Alert,TaskForceUUID) & Alert=ID <-
        .send(actuator,achieve,redAlert);       .send(actuator,achieve,infoLCD("Evacuate!!!!")).
11  +!alert(ID,atention): taskForce(Alert,TaskForceUUID) & Alert=ID <-
        .send(actuator,achieve,yellowAlert);    .send(actuator,achieve,infoLCD("Atention!!!")).
```

Code 1.3. Implementation of agent *leader* in house MAS.

The agent *forecaster* has an initial goal and two plans:

- The initial goal (*start*) triggers the first plan for getting data from a public service (*.inmetGovBrCheck* — internal action proposed in this paper).
- The second plan (*inmetAlert*) is activated if a new alert is received, then the agent should to inform the agent leader about the alert.

```
    !start. /* initial goal */
2   +!start: myCityIBGE(COD) <-  .inmetGovBrCheck("https://apiprevmet3.inmet.gov.br/avisos/rss",COD).
    +inmetAlert(Alert,Event,Serverity,Certainty,Time,Response,Detail,Info,Url):Time=rightNow&Serverity=severe <-
4       .send(leader,tell,weatherEvent(Alert,Event,Serverity)).
```

Code 1.4. Implementation of agent *forecaster* in house MAS.

The agent *actuator* has an initial goal and six plans:

- The initial goal triggers the first plan (*start*) to opening the perceptions and getting information from low-end IoT device connected on the serial port (*.port* and *.percepts* — are specific *ARGO* internal actions).
- The second plan (*infoTaskForce*) requests to *telephonist* to forward information about the sensors directly to the agent *assistant* in the coalition.
- The other plans (lines 5-8) deal with sending commands to low-end IoT device, acting in the physical world (*.act* — specific *ARGO* internal action).

```
1   !start.
    +!start <-  .port(ttyUSB0); .percepts(open).
3   +!infoTaskForce:  rainLast24hrs(RainStatus) &  waitingInstructions(Alert,TF,OurUUID)<-
        .send(telephonist,achieve,sendExternalMessage(TF,assistant,tell,rainLast24hrs(OurUUID,RainStatus))).
5   +!infoLCD(Message) <- .act(Message).
    +!yellowAlert <- .act(yellowAlert).
7   +!greenAlert <- .act(greenAlert).
    +!redAlert <- .act(redAlert).
```

Code 1.5. Implementation of agent *actuator* in house MAS.

- **The *telephonists* reasoning:** All MAS in scenario has a *Communicator* agent (named *telephonist*) with an initial goal and five plans:

- The initial goal triggers the first plan (*connect*) to connecting with the IoT network (*.connectCN* — specific *Communicator* internal action).
- The second plan (*transportAnAgentTo*) migrate a specific agent from the MAS to an open MAS (*.moveOut* — specific *Communicator* internal action).
- The third plan (*internalUnicastMessage*) forwards a message from another MAS to the internal destination agent in the MAS.
- The last two plans (*mas2masMessage* and *sendExternalMessage*) forward a message to the agent *telephonist* in another MAS (*.sendOut* — specific *Communicator* internal action).

```
!connect. /* initial goal */
+!connect: myIotAddress(UUID) <- .connectCN("skynet.chon.group",5500,UUID).
+!tranportAnAgentTo(Agent,Destination)<-  .moveOut(UUID,mutualism,Destination).
+!internalUnicastMessage(Destination,Force,Message)[source(X)]<-
    .send(Destination,Force,Message[source(X)]).
+!mas2masMessage(UUID,Force,Message)<-    .sendOut(UUID,Force,Message).
+!sendExternalMessage(MAS,Receiver,Force,Message)[source(Source)]<-
    .sendOut(MAS, achieve, internalUnicastMessage(Receiver,Force,Message[source(Source)])).
```

Code 1.6. Implementation of agent *telephonist* in all MAS.

4.2 Reproducibility

This paper provides two new internal actions for Jason agents. The former (*.create_mas*) receives three parameters: a .zip file with a MAS project that will be executed, a term (*gui*|*cli*) determining if the coalition will run in a graphical or text environment, and an address into the IoT network to inform the successful execution. The second (*.inmetGovBrCheck*) receives the URL public service of the ALERT-AS and a COD to filter the information by location. Aiming to ensure the reproducibility of this work, the source code, the scenario implementation, the firmware code, and a video demonstration are available[3].

5 CONCLUSION

The development of a communication architecture based on MAS for embedded systems represents a significant contribution to the resilience of cities in critical moments. Our proposal successfully filled the gap in communication between embedded systems. The agents are designed to collect, process, and transmit critical information in a distributed manner effectively during field testing. The results prove that MAS can efficiently coordinate the emergency response during crisis events. Achieving real-time cooperation between embedded systems is essential for smart cities functioning in crises. It not only improves citizen security but also optimizes the use of resources and coordination of actions at critical moments. Our research represents an important step towards resilient and safer cities. Responding to crisis events is critical to protecting life and property, and our agent-based approach offers a low-cost, viable, and effective solution to this challenge. As urbanization grows, it is imperative to provide solutions to ensure that smart cities remain resilient and prepared to face any challenge.

Although our research has contributed to improving the coordination of embedded systems in smart cities during crisis events, it is important to recognize that there are limitations to security and communication infrastructure. Real-time cooperation between embedded systems raises security concerns, especially regarding data protection and cyberattack vulnerability. Considering the communication infrastructure, the effectiveness of the architecture may depend on the availability of reliable communication technology, such as high-speed communication networks. The architecture's applicability may be limited in rural areas or regions with underdeveloped communications infrastructure.

[3] `https://papers.chon.group/PAAMS/2024/DARC/`

There are promising directions for future work concerning crisis events and MAS. Improving the agents' autonomy and adaptability allows more sophisticated decision-making during crises. Additionally, exploring advanced cybersecurity approaches to protect MAS from potential threats and attacks becomes vital as systems interconnection increases. Integrating autonomous vehicles and drones into the architecture of MAS represents another promising research front, with the potential to enable an even faster and comprehensive response to crises.

Furthermore, to deal with specific types of disasters, such as wildfires, floods, or pandemics, emerges as an alternative area of research. Studies that address community acceptance of technology and involve citizens in data generation and decision-making can also improve the architecture's effectiveness. As urbanization and challenges associated with crisis events evolve, future research can contribute to even more effective and comprehensive solutions, promoting the security and resilience of smart cities worldwide.

References

1. Blaikie, P., Cannon, T., Davis, I., Wisner, B.: At risk: natural hazards, people's vulnerability and disasters. Routledge (2014). https://doi.org/10.4324/9780203714775
2. Bordini, R.H., Hübner, J.F.: BDI Agent Programming in AgentSpeak Using Jason. In: Computational Logic in Multi-Agent Systems. pp. 143–164. Springer, Berlin, Heidelberg (2006). https://doi.org/10.1007/11750734_9
3. Bordini, R.H., Hübner, J.F., Wooldridge, M.: Programming multi-agent systems in AgentSpeak using Jason. John Wiley & Sons (2007)
4. Brandão, F.C., Lima, M.A.T., Pantoja, C.E., Zahn, J., Viterbo, J.: Engineering Approaches for Programming Agent-Based IoT Objects Using the Resource Management Architecture. Sensors **21**(23) (2021). https://doi.org/10.3390/s21238110
5. Bratman, M.E., Israel, D.J., Pollack, M.E.: Plans and resource-bounded practical reasoning. Computational Intelligence **4**(3), 349–355 (Sep 1988). https://doi.org/10.1111/j.1467-8640.1988.tb00284.x
6. Brouwer, T.: Potential of twitter derived flood maps: comparing interpolation methods and assesing uncertainties (2016), `http://essay.utwente.nl/71007/`
7. Dottori, F., Szewczyk, W., Ciscar, J.C., Zhao, F., Alfieri, L., Hirabayashi, Y., Bianchi, A., Mongelli, I., Frieler, K., Betts, R.A., et al.: Increased human and economic losses from river flooding with anthropogenic warming. Nature Climate Change **8**(9), 781–786 (2018). https://doi.org/10.1038/s41558-018-0257-z
8. Endler, M., Baptista, G., Silva, L.D., Vasconcelos, R., Malcher, M., Pantoja, V., Pinheiro, V., Viterbo, J.: Contextnet: Context reasoning and sharing middleware for large-scale pervasive collaboration and social networking. PDT '11, ACM, New York, NY, USA (2011). https://doi.org/10.1145/2088960.2088962
9. Lazarin, N.M., Pantoja, C.E.: A robotic-agent platform for embedding software agents using raspberry pi and arduino boards. In: Proceedings of the WESAAC 2015. pp. 13–20. UFF, Niterói (2015), `http://www2.ic.uff.br/~wesaac2015/Proceedings-WESAAC-2015.pdf`
10. Lazarin, N.M., Pantoja, C.E., Viterbo, J.: Dealing with the unpredictability of physical resources in real-world multi-agent systems. In: Rocha, A.P., Steels, L., van den Herik, J. (eds.) Agents and Artificial Intelligence. pp. 48–71. Springer Nature Switzerland, Cham (2024)

11. Marouane, E.M.: Towards a real time distributed flood early warning system. IJACSA **12**(1) (2021). https://doi.org/10.14569/IJACSA.2021.0120162
12. Matsuki, A., Hatayama, M.: Identification of issues in disaster response to flooding, focusing on the time continuity between residents' evacuation and rescue activities. IJDRR **95** (2023). https://doi.org/10.1016/j.ijdrr.2023.103841
13. Mishra, A., Alnahit, A., Campbell, B.: Impact of land uses, drought, flood, wildfire, and cascading events on water quality and microbial communities. Journal of Hydrology **596**, 125707 (2021). https://doi.org/10.1016/j.jhydrol.2020.125707
14. Mosavi, A., Ozturk, P., Chau, K.w.: Flood prediction using machine learning models: Literature review. Water **10**(11) (2018). https://doi.org/10.3390/w10111536
15. Ojo, M.O., Giordano, S., Procissi, G., Seitanidis, I.N.: A Review of Low-End, Middle-End, and High-End Iot Devices. IEEE Access **6**, 70528–70554 (2018). https://doi.org/10.1109/ACCESS.2018.2879615
16. Pantoja, C.E., Jesus, V.S.d., Lazarin, N.M., Viterbo, J.: A Spin-off Version of Jason for IoT and Embedded Multi-Agent Systems. In: Intelligent Systems. Springer Nature Switzerland, Cham (2023). https://doi.org/10.1007/978-3-031-45368-7_25
17. Pantoja, C.E., Stabile, M.F., Lazarin, N.M., Sichman, J.S.: ARGO: An Extended Jason Architecture that Facilitates Embedded Robotic Agents Programming. In: Engineering Multi-Agent Systems. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-50983-9_8
18. Rafanelli, A., Costantini, S., De Gasperis, G.: A multi-agent-system framework for flooding events. In: Proceedings of the WOA 2022. pp. 142–151. Genova (2022), `https://ceur-ws.org/Vol-3261/paper11.pdf`
19. Rakotoarisoa, M.M., Reulier, R., Delahaye, D.: Agent-based modelling of the evolution of hydro-sedimentary connectivity: The case of flash floods on arable plateaus. Appl. Sci. **13**(5) (2023). https://doi.org/10.3390/app13052967
20. Sasaki, J., Kitsuya, M.: Development and evaluation of regional information sharing system (RISS) for disaster risk reduction. Inf Syst Front **23**, 1203–1211 (2021). https://doi.org/10.1007/s10796-020-10076-7
21. Sichman, J.S.: DEPINT: Dependence-Based Coalition Formation in an Open Multi-Agent Scenario. Journal of Artificial Societies and Social Simulation **1**(2) (1998), `https://www.jasss.org/1/2/3.html`
22. Simmonds, J., Gómez, J.A., Ledezma, A.: The role of agent-based modeling and multi-agent systems in flood-based hydrological problems: a brief review. Journal of Water and Climate Change **11**(4) (2020). https://doi.org/10.2166/wcc.2019.108
23. Souza de Jesus., V., Pantoja., C.E., Manoel., F., Alves., G.V., Viterbo., J., Bezerra., E.: Bio-Inspired Protocols for Embodied Multi-Agent Systems. In: Proceedings of the ICAART 2021. pp. 312–320. SciTePress (2021). https://doi.org/10.5220/0010257803120320
24. Tingsanchali, T.: Urban flood disaster management. Procedia Engineering **32**, 25–37 (2012). https://doi.org/10.1016/j.proeng.2012.01.1233, iSEEC
25. Vu, T.M., Mishra, A.K.: Nonstationary frequency analysis of the recent extreme precipitation events in the United States. Journal of Hydrology **575**, 999–1010 (2019). https://doi.org/10.1016/j.jhydrol.2019.05.090
26. Wooldridge, M.J.: An Introduction to MultiAgent Systems. John Wiley & Sons, Chichester, U.K, 2nd edn. (2009)
27. Yaning, G., Qianwen, W.: Analysis of collaborative co-governance path of public crisis emergency management in an all-media environment. In: Proccedings of the ICMSSE 2021 (2021). https://doi.org/10.1109/ICMSSE53595.2021.00057