CoMuRoS: A LLM-Driven Hierarchical Architecture for Adaptive Multi-Robot Collaboration

Suraj Borate[1], Bhavish Rai B[2], Vipul Pardeshi[3] and Madhu Vadali[4]

***Abstract*— Dynamic environments require multi-robot teams to adapt to disruptive events by finding collaborative plans to achieve the goals. We introduce CoMuRoS (Collaborative Multi-Robot System)[1], an LLM-powered hierarchical architecture for heterogeneous robots that enables zero-shot task classification, allocation, and event-triggered replanning. A central task manager LLM processes natural language goals, categorizes tasks (independent, sequential, coordinated, infeasible) and assigns them based on capabilities and context. Local robot LLMs generate executable code from primitives, with perception classifying events to initiate replanning for recovery or human aid. Hardware tests validate adaptation to failures (e.g.,collaborative object recovery: 9/10 success and emergent human-multi-robot collaboration (5/5)). Simulations demonstrate architecture's applicability for understanding and responding to user intentions in health care and disaster relief scenario. A 22-scenario benchmark demonstrates generalization (correctness: 0.91 with Grok-3). CoMuRoS advances reasoning and planning in dynamic environments by blending centralized deliberation with decentralized execution.**

I. INTRODUCTION

In dynamic environments, heterogenous multi-robot systems must adapt to failures, feedback or user-intent changes using their diverse capabilities. Large Language Models (LLMs) enable robots to interpret complex instructions via context, common sense, and general knowledge. However, most LLM-based planners rely on complete information in static environment which reduces possibility of recovery in case of a disruption. This paper presents CoMuRoS, a hierarchical architecture for zero-shot, event-driven replanning in heterogeneous multi-robot teams, inspired by human organizational structures where managers allocate tasks and agents report issues. CoMuRoS uses a Task Manager LLM to interpret goals, classify tasks (independent, sequential, coordinated, infeasible), and assign them based on capabilities and context. It supports replanning which enables teammate aid, resumption, and need based human collaboration. Robots use local LLMs to generate executable code, proactively notifying the Task Manager of relevant events. Effectiveness is shown through hardware experiments and simulations. 22-scenario textual benchmark is curated and evaluated to highlight generalization (correctness: 0.91 with Grok-3).

*This work was not supported by any organization

[1]Suraj Borate is a PhD scholar in the Department of Mechanical Engineering, IIT Gandhinagar, Gujarat, India. surajb@iitgn.ac.in

[2]Bhavish Rai B is B.Tech graduate from Sahyadri College of Engineering and Management, Adyar, Karnataka, India. bhavishraib@gmail.com

[3]Vipul Pardeshi is with Vishwakarma Institute of Information Technology, Pune, Maharashtra, India . pardeshivipul18@gmail.com

[4]Madhu Vadali is an Associate Professor in the Department of Mechanical Engineering, IIT Gandhinagar, Gujarat, India. madhu.vadali@iitgn.ac.in

II. RELATED WORK

Robot Foundation Models such as RT-1/RT-2 [2], [3], OpenVLA [4], and GR00T [5] map observations to actions for individual robots, excelling in manipulation tasks but still lack mechanisms for multi-agent coordination. These trained policies often fail on long-horizon tasks and event-driven adaptations, where LLM-based planners show greater flexibility. Classical symbolic planners (e.g., the Planning Domain Definition Language, PDDL [6]) are effective in fully known environments but degrade under partial observability and dynamic changes. LLMs have enabled zero-shot planning for single-robot settings [7], [8]. Recent LLM-based multi-robot frameworks, such as SMART-LLM [9] (Python-based, no ROS2/hierarchy) and COHERENT [10] (action selection without code generation), overlook runtime event handling. Decentralized approaches like CoELA [11] face scalability challenges in terms of token efficiency and decision making time, while hybrid methods such as DART-LLM [12] impose rigid Directed Acyclic Graph (DAG) dependencies. In contrast, CoMuRoS [1] adopts a hierarchical design with a central LLM for planning and replanning, delegating execution to local policies or ROS2 nodes, while supporting event-driven replanning, event detection and classification, and seamoless human interaction. This distinguishes it from DAG-based systems [12], which remain brittle under unforeseen events.

III. ARCHITECTURE

CoMuRoS adopts a hierarchical design inspired by organizational teams. User-specified objectives are processed by the Task Manager, which allocates tasks to agents based on their capabilities and constraints. Agents then function as team members, carrying out the assignments while monitoring their environment and reporting significant events to the Task Manager. The resulting architecture embodies this manager–teammate model. The architecture (Fig. 1) embodies a hybrid paradigm: centralized high-level planning by the Task Manager LLM and decentralized execution and event handling by robot LLMs.

A. Task Manager LLM

The Task Manager LLM is responsible for three key functions: task classification, task allocation, and task replanning in response to significant events. It first classifies user-defined tasks as *sequential*, *independent*, *coordinated*, or *infeasible*.

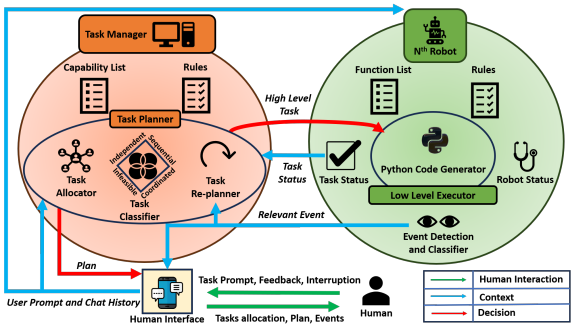


Fig. 1: Architectural Diagram of CoMuRoS.

Classification decides team formation and schedules communicating tasks to robots. Tasks are then allocated to robots according to their morphological capabilities, operational constraints, and available skills. When either a robot or a human reports an update that affects execution, the Task Manager performs replanning to ensure task completion.

Replanning uses the context of robot statuses, task statuses and chat history, which maintains a record of prior commands, and reported events. Prompt engineering(temperature = 0.5) is used to harness the commonsense reasoning and contextual understanding of LLMs, enabling them to interpret user instructions, classify tasks and events, and allocate work according to robot capabilities.

The Task Manager prompt is divided into two parts. The static prompt encodes rules, definitions, formats, and procedural logic, effectively serving as the "brain" of the Task Manager by storing knowledge of task categories, allocation policies, and replanning procedures. The dynamic prompt integrates scenario-specific information such as user commands, robot and task statuses, chat history, and detected events. Scenario configuration files additionally specify robot capabilities and contextual rules. Task status may be reported as `COMPLETED`, `IN PROGRESS`, or `INTERRUPTED`. During replanning, only tasks not marked as `COMPLETED` are reconsidered. Robot status is particularly critical for sequential tasks; for instance, a robotic arm with limited reach can place an object on a quadruped, only if the quadruped is sitting. The system also supports human–robot collaboration. If humans are included in the dynamic prompt, the Task Manager can assign tasks to them in cases where no robot is capable of execution.

B. Robot LLM

The proposed architecture is designed to be robot-agnostic, supporting heterogeneous platforms. Each robot receives high-level natural language commands from the Task Manager, which guarantees feasibility relative to its constraints. As illustrated in Fig. 1, every robot maintains a library of execution functions, each annotated with its description, inputs, outputs, and intended use cases. These functions may correspond to reinforcement learning policies, imitation learning models, ROS nodes, or conventional Python code. Upon receiving a command, the robot's LLM decomposes it, selects appropriate functions, and synthesizes Python code to produce an executable low-level plan.

Robots do not differentiate between sequential and independent tasks. Each processes one command at a time, and upon completion, updates the task status to `COMPLETED`. Execution functions are expected to update both the task status and the robot status, either through sensor feedback or open-loop logic. Robot status may include descriptors (e.g., "sitting," "standing") or coordinates such as $(4, 0)$.

In parallel with execution, robots continuously monitor their environment to detect and classify events. Events can be identified through onboard sensors or external sources such as CCTV. An LLM determines whether each detected event is relevant (e.g., influencing task feasibility or progress) or irrelevant (e.g., "It stopped raining" during an indoor cleaning task). Relevant events are reported to the Task Manager, which then initiates replanning. Event detection can be implemented using vision–language models, action recognition pipelines, or image processing methods.

C. Human Interface

A chat-based interface facilitates continuous natural language communication between humans and the multi-robot system (Fig. 2). Through this interface, the user may issue new commands, provide contextual information, interrupt ongoing tasks, or revise intentions at any point during execution.

The dialogue is color-coded for clarity: user commands appear in blue, Task Manager allocations in orange, and robot-detected events in green, all displayed in chronological order. The preserved chat history allows the system to interpret context-sensitive instructions such as "repeat the earlier task" or "pick it up again." This also ensures transparency in replanning, since the triggering event and the corresponding updated plan are shown together.

Currently, the interface supports typed commands, but it can be extended to speech input and multilingual interaction depending on the underlying LLM. In this way, the system supports fluid and adaptive collaboration between humans and robots.

D. Evaluation Metrics

A textual dataset comprising 22 diverse scenarios, each containing three tasks, was constructed to evaluate high level planning of CoMuRoS. Each scenario is paired with a configuration file specifying the list of robots, their capabilities, and scenario-specific rules, along with three human-provided task prompts. Evaluation is performed using the following metrics: **Task Allocation (TA)**, assigned a value of 1 if tasks are allocated correctly according to both robot capabilities and user instructions, and 0 otherwise; **Task Classification (TC)**, assigned a value of 1 if the classification is feasible under the corresponding allocation, and 0 otherwise (noting that certain tasks may validly be executed either sequentially or independently); **Intersection-over-Union (IoU)**, the standard metric that quantifies the overlap between the Task Manager's generated plan and the ground truth on a scale of 0–1; **Executability (Exec)**, defined as the proportion of allocations that are feasible given robot capabilities and

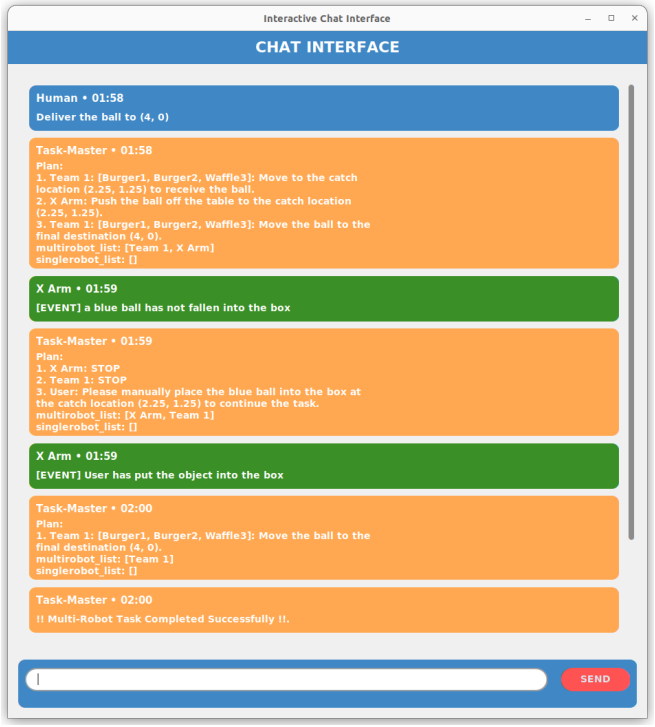


Fig. 2: Chat Interface for human robot collaboration.

constraints, normalized to a range of 0–1; and **Correctness**, equal to 1 if an expert evaluator determines that the plan would succeed, and 0 otherwise.

IV. Experiments and Results

Hardware. Experiments were conducted on TurtleBot platforms, a Unitree Go2 quadruped, and OpenManipulator-X arms to evaluate CoMuRoS in task allocation, event detection, classification, and event-driven replanning with onboard sensing. Two sets of hardware demonstrations were performed: (1) Multi-robot collaboration with replanning, and (2) Human–formation collaboration. https://youtu.be/9zOm0uG16YQ .

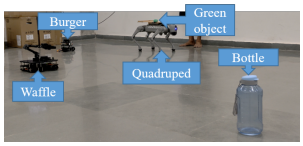
In the multi-robot recovery scenario, a Go2, TurtleBot Burger, and TurtleBot Waffle-pi with an arm (Waffle) were deployed. The Waffle was instructed to approach a bottle (local LLM composes Python code from functions such as `find(object)` and `reach(object)`), while the Go2 was tasked with carrying a green object to $(3, 0)$. During execution, the object was deliberately dropped by the human for disruption (Fig. 3b). CoMuRoS detected the event online, classified it as relevant, and initiated event-driven replanning. The Task Manager reallocated sequential roles while accounting for physical constraints: the task of helping the quadruped was given to Waffle due to its manipulation ability, also plan ensure quadruped sits because Waffle's manipulator could not reach a standing Go2, for object transfer. This shows task and robot status monitoring. After the collaborative multi-robot recovery of the fallen object, Waffle resumed its original task. In ten trials (using GPT-4o), the success rate was **9/10**, with one failure due to an incorrect replanning sequence.

In coordinated object transport experiment, three mobile robots (one Waffle and two Burgers) carried a box, while an OpenManipulator-X arm (X-arm) manipulated a blue ball on a table. The user command - "Deliver the blue ball to (4,0)," was classified as sequential and coordinated. The Task Manager generated a plan correctly classifying this task as coordinated task he robot formation transported the box to the catch point, the X-arm pushed the ball into the box, and its camera verified success. If ball falls inside box, the formation delivered the box to the goal, achieving a success rate of **8/8**.https://youtu.be/UtOMZBV_mUQ In the failure case (Fig. 4), the ball fell outside the box. The X-arm's camera detected this and reported it to both the Task Manager and the human interface, triggering replanning. Since no robot could recover the ball, the task was reassigned to the human user, who placed it in the box. The formation then completed delivery. Detection combined OwlViT-based vision and GPT-4.1 VQA. This emergent human-multirobot collaboration experiment achieved a success rate of **5/5**, showing the ability of CoMuRoS to engage humans when tasks exceed capabilities of robots. The user interaction logs for this experiment are shown in 2.

Simulations. To highlight applicability to broader domains, two simulated scenarios were implemented in Gazebo: disaster relief and hospital scenario. In the disaster relief scenario, a Go2 quadruped and a drone collaboratively searched for survivors and delivered first aid, with CoMuRoS dynamically replanning to prioritize human feedback on urgent cases.https://youtu.be/RCi28zFyGT4 . In the hospital scenario (Fig. 5), a Go2 quadruped and two UR5 arms collaborated to deliver food. The user prompt "I am hungry" was classified as sequential: the chef arm loaded the plate, the Go2 carried it, and the helper arm served it. When the user interrupted with "I don't want it anymore," this was identified as a relevant event, and replanning redirected the Go2 to return the food. These results demonstrate the ability of CoMuRoS to interpret indirect human intent and act accordingly.

Benchmarking. The high-level planning ability of CoMuRoS was evaluated using a curated textual benchmark dataset of 22 scenarios (three tasks each), tested across eight LLMs. As shown in Fig. 6, CoMuRoS achieved the highest performance using Grok-3, with an average correctness of 0.91 across all scenarios. Additional metrics for Grok-3 include: Task Allocation (TA) = 0.96, Task Classification (TC) = 0.96, IoU = 0.97, and Executability = 0.98.

To evaluate replanning, a second dataset of 5 scenarios (3 tasks each) was constructed, including an initial user command, and a disruptive event. The replanning response to a simulated event was scored using the same metrics, when the initial plan was correct and the event relevant. As shown in Fig. 7b, CoMuRoS achieved perfect correctness (1.0) across all 5 scenarios, iterations, and tasks with Grok-3.Finally, robustness to linguistic variation was evaluated on semantically identical commands with different phrasings. Results in Fig. 8 show that CoMuRoS maintained consistently high correctness across GPT-4, GPT-4o, and Grok-3,



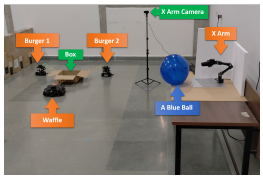
(a) User instructs waffle to go to bottle and Quadruped to carry green object to (4,0)

(b) Disruption due to object drop.

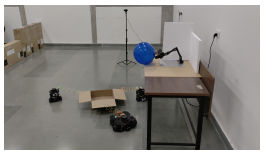
(c) Waffle fetching and placing the green object on Go2.

(d) After helping waffle completes its original task.

Fig. 3: Demonstration of event-driven replanning, incomplete task resumption, and emergence of cooperation where robots assist one another. https://youtu.be/W_0TnEH_UwU



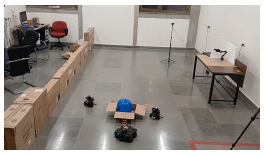
(a) Initial setup.



(b) Ball pushed by X-arm.

(c) Ball falls outside.

(d) Human intervention.



(e) Formation completes delivery.

Fig. 4: Human–formation collaboration experiment: (a) initial setup, (b) ball pushed by X-arm, (c) failure case with ball outside, (d) human places the ball inside, and (e) formation delivers the ball successfully. https://youtu.be/7QZDq5MSq1A

(a) Patient refuses food.

(b) Quadruped returns plate.

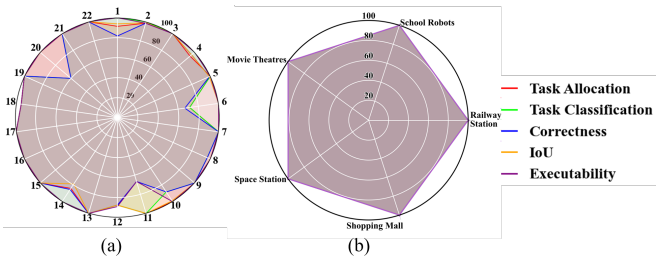
Fig. 5: Hospital scenario: anytime human interruption and event-driven replanning. https://youtu.be/qp81etRRkfg.

Fig. 6: Correctness scores across 22 scenarios for different LLMs (GPT-4, GPT-4o, GPT-4.1, Grok-3, DeepSeek-r1:7b, Llama-3:70b, DeepSeek-r1:32b, Qwen-3:32b). Scenarios are diverse 1-22 ranging from reforestation to asteroid mining. Not listed for brevity.

highlighting resilience to natural variations in user input. The complete set of videos, interaction logs, code, and textual dataset is available at CoMuRoS.github.io

V. CONCLUSION

This work presents a hierarchical multi-robot architecture, CoMuRoS, that combines Large Language Models for enabling proactive and adaptive teamwork. The task



(a) (b)

Fig. 7: (a) Performance across 22 scenarios for Grok-3, averaged over all tasks and iterations. (b) Replanning evaluation across 5 scenarios with Grok-3, showing 1.0 correctness across all trials.

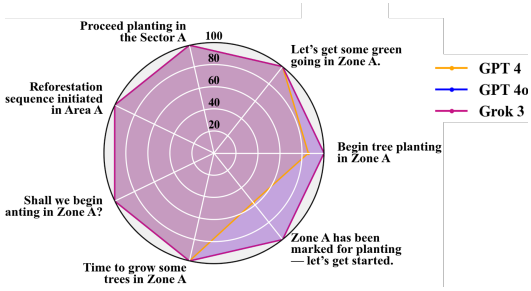


Fig. 8: Correctness scores for semantically identical commands, demonstrating robustness to command phrasing.

manager redistributes tasks in response to events, inspired by human teams where members can signal issues and managers adapt roles. Hardware trials on heterogeneous robots and simulations show high success rates. The architecture showed generalization in wide range of scenarios. The modular approach, with separate Task Manager logic and adaptable scenario configurations, ensured consistent success across datasets, simulations, and hardware experiments. **Limitations:** Replanning currently assumes task completion is feasible after failures; future directions include reasoning about unrecoverable failures, operation in sub-teams and pursuing partial completion to minimize loss.

REFERENCES

[1] S. Borate, B. Rai B, V. Pardeshi, and M. Vadali, "Comuros: Adaptive hierarchical planning for multi-robot teams in dynamic environments using llms," 2025, manuscript submitted to an international robotics conference.

[2] A. Brohan, N. Brown, J. Carbajal, Y. Chebotar, J. Dabis, C. Finn, K. Gopalakrishnan, K. Hausman, A. Herzog, J. Hsu *et al.*, "Rt-1: Robotics transformer for real-world control at scale," *arXiv preprint arXiv:2212.06817*, 2022.

[3] B. Zitkovich, T. Yu, S. Xu, P. Xu, T. Xiao, F. Xia, J. Wu, P. Wohlhart, S. Welker, A. Wahid *et al.*, "Rt-2: Vision-language-action models transfer web knowledge to robotic control," in *Conference on Robot Learning*. PMLR, 2023, pp. 2165–2183.

[4] M. J. Kim, K. Pertsch, S. Karamcheti, T. Xiao, A. Balakrishna, S. Nair, R. Rafailov, E. Foster, G. Lam, P. Sanketi *et al.*, "Openvla: An open-source vision-language-action model," *arXiv preprint arXiv:2406.09246*, 2024.

[5] J. Bjorck, F. Castañeda, N. Cherniadev, X. Da, R. Ding, L. Fan, Y. Fang, D. Fox, F. Hu, S. Huang *et al.*, "Gr00t n1: An open foundation model for generalist humanoid robots," *arXiv preprint arXiv:2503.14734*, 2025.

[6] C. Aeronautiques, A. Howe, C. Knoblock, I. D. McDermott, A. Ram, M. Veloso, D. Weld, D. W. Sri, A. Barrett, D. Christianson *et al.*, "Pddl—the planning domain definition language," *Technical Report, Tech. Rep.*, 1998.

[7] W. Huang, P. Abbeel, D. Pathak, and I. Mordatch, "Language models as zero-shot planners: Extracting actionable knowledge for embodied agents," in *International conference on machine learning*. PMLR, 2022, pp. 9118–9147.

[8] D. Shah, B. Osiński, S. Levine *et al.*, "Lm-nav: Robotic navigation with large pre-trained models of language, vision, and action," in *Conference on robot learning*. PMLR, 2023, pp. 492–504.

[9] S. S. Kannan, V. L. Venkatesh, and B.-C. Min, "Smart-llm: Smart multi-agent robot task planning using large language models," in *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2024, pp. 12 140–12 147.

[10] K. Liu, Z. Tang, D. Wang, Z. Wang, X. Li, and B. Zhao, "Coherent: Collaboration of heterogeneous multi-robot system with large language models," *arXiv preprint arXiv:2409.15146*, 2024.

[11] Y. Chen, J. Arkin, Y. Zhang, N. Roy, and C. Fan, "Scalable multi-robot collaboration with large language models: Centralized or decentralized systems?" in *2024 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2024, pp. 4311–4317.

[12] Y. Wang, R. Xiao, J. Y. L. Kasahara, R. Yajima, K. Nagatani, A. Yamashita, and H. Asama, "Dart-llm: Dependency-aware multi-robot task decomposition and execution using large language models," *arXiv preprint arXiv:2411.09022*, 2024.