Multi-Agent Campus Tour System Using ROS2-foxy and CrewAI

Adarsh Raj Shrivastava (B21AI003)
Prakhar Gupta (B21AI027)

Contents

1 Introduction

This report describes the multi-agent system designed to facilitate campus tours using ROS2-foxy and the CrewAI framework. Agents include: Campus Incharge (CI), Building Incharge (BI), and Visitor agent. They interact with each other and the environment to guide the visitor agent from the campus entrance to their required room within a building. The system uses `EscortRequest.srv` and `NavigationRequest.srv` services for agent communication, and the campus map is visualized using RViz.

2 System Architecture

The system is implemented in ROS2-foxy, with each node being a separate package. Following are the packages implemented:

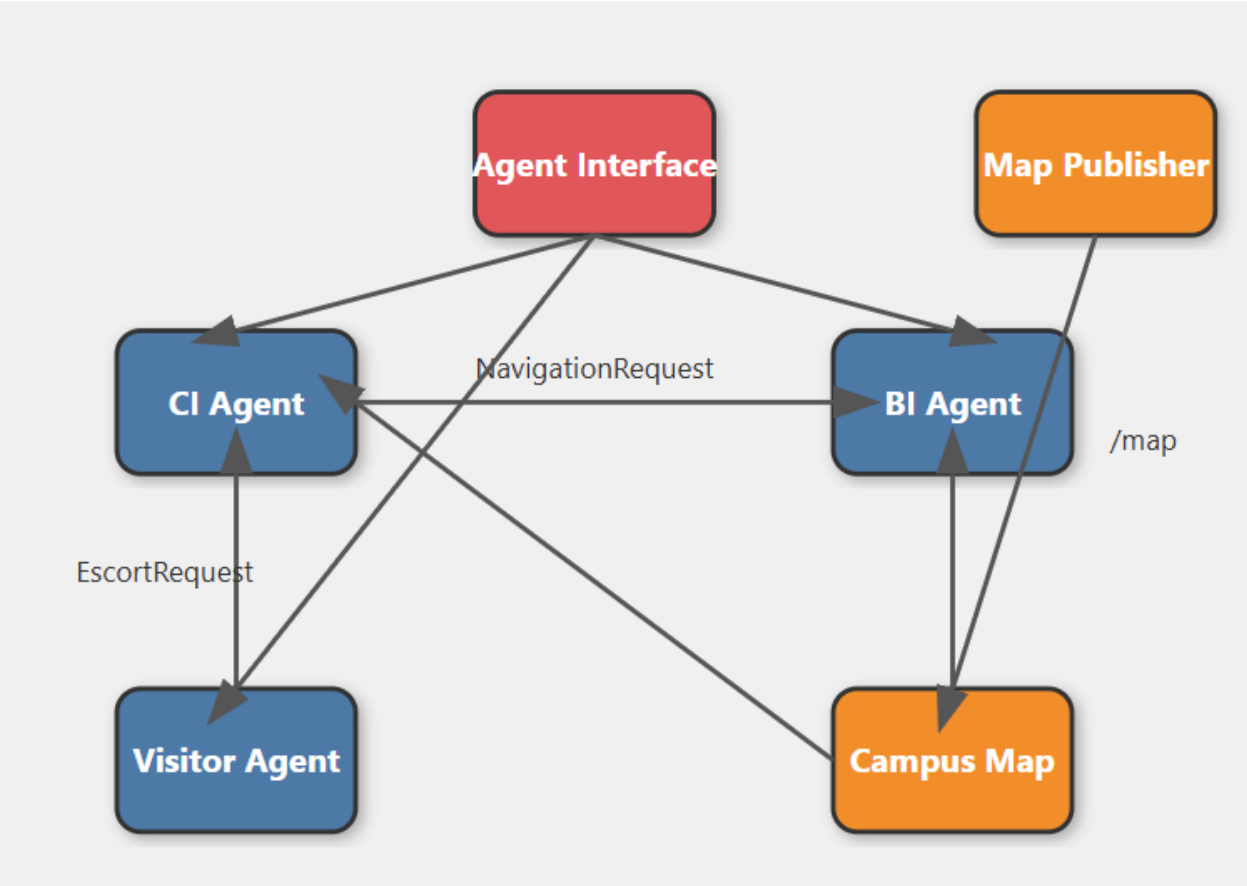


Figure 1: Architecture

- **ci_agent**: Campus Incharge agent package. Responsible for handling visitor agent requests and escorting them to the required room within the building.
- **bi_agent**: Building Incharge agent package. Responsible for handling navigation requests from the campus incharge agent and providing paths to required rooms within the building.
- **visitor_agent**: Visitor agent package. Responsible for sending escort requests to the campus incharge agent.
- **campus_map**: Campus map package. Stores the map of the campus including buildings and rooms.

- **map_publisher**: Map publisher package. Publishes the campus map to the `/map` topic.
- **agent_interface**: Agent interface package. Contains service definitions for communication between agents.

Time required to build the packages:

```
Starting >>> agent_interfaces
Starting >>> bi_agent
Starting >>> campus_map
Starting >>> ci_agent
Finished <<< agent_interfaces [2.48s]
Starting >>> map_publisher
Finished <<< ci_agent [3.78s]
Starting >>> visitor_agent
Finished <<< campus_map [4.34s]
Finished <<< bi_agent [4.37s]
Finished <<< map_publisher [2.44s]
Finished <<< visitor_agent [1.77s]

Summary: 6 packages finished [6.05s]
```

3 Agent Communication Protocol

The agents communicate with each other using the following services:

3.1 EscortRequest.srv

Service used by the visitor agent to request escort from the campus incharge agent.

```
string visitor_id
string building_location
string room_number
bool oosrequest
int32 oosmeettime
---
bool success
string message
string path
```

3.2 NavigationRequest.srv

Service used by the campus incharge agent to request a path to the required room within the building from the building incharge agent.

```
string visitor_id
string host_location
bool ossrequest
int32 ossmeettime
---
bool authorized
string path
string denial_reason
int32 osswaittime
```

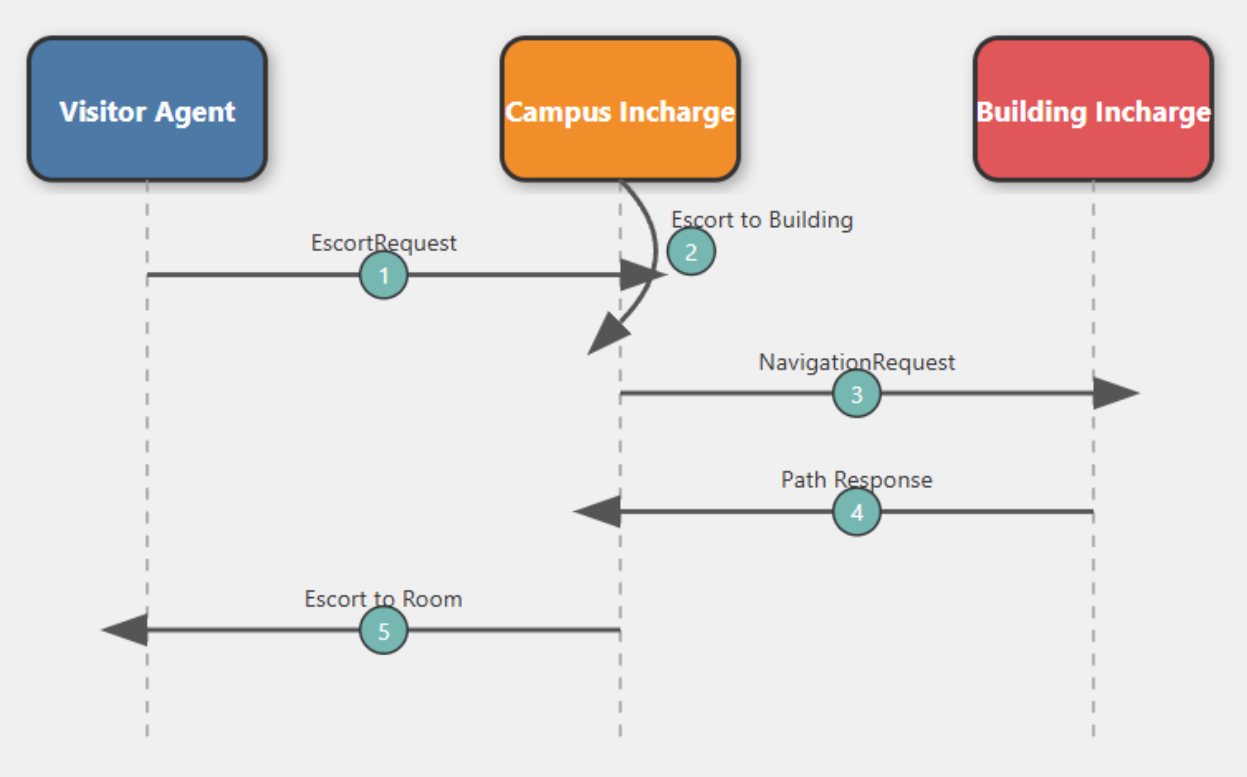


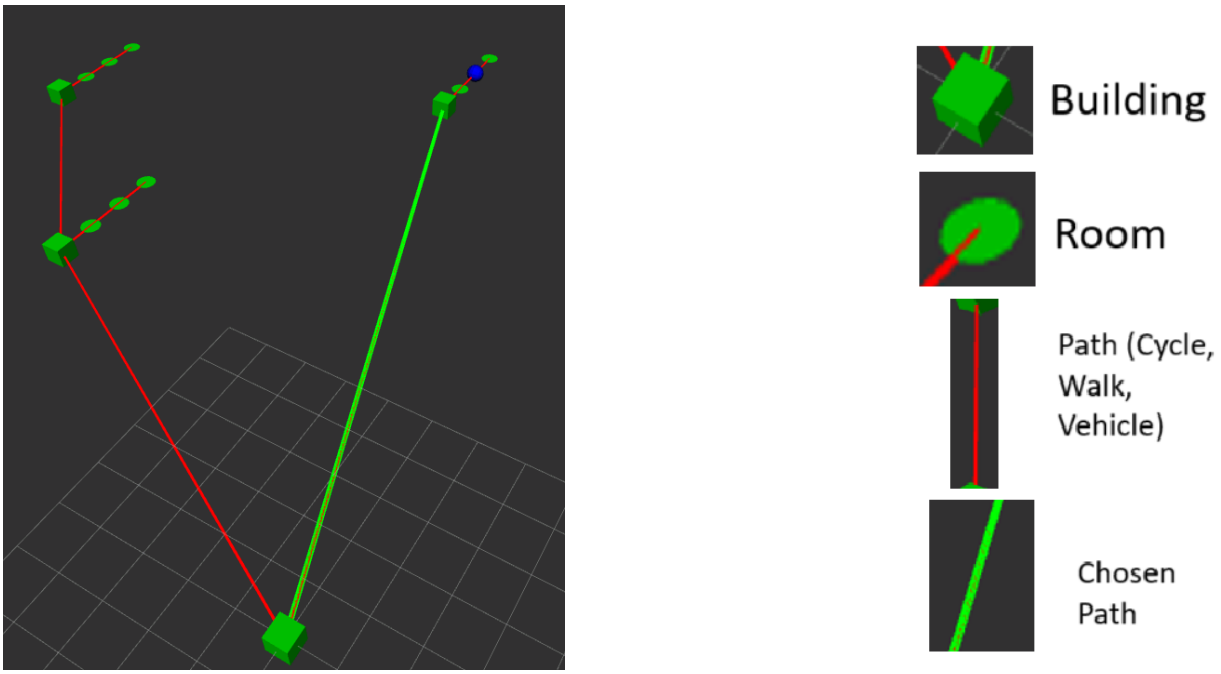
Figure 2: Message Flow

4 Message Flow

1. Visitor agent sends an escort request to the campus incharge (CI) agent.
   - The CI agent meets visitors at the campus entrance.
   - This step is implemented using the `EscortRequest.srv` service.
2. CI agent receives the request and escorts the visitor agent to the required building.
   - This interaction is handled by the CI agent as part of its responsibilities.
   - The escort process is visualized in RViz, with the CI agent represented in dark blue and the visitor agent in light blue.
3. CI agent sends a navigation request to the building incharge (BI) agent.
   - This communication occurs once inside the building.
   - The request is implemented using the `NavigationRequest.srv` service.
4. BI agent receives the request and provides the path to the required room within the building.
   - The BI agent processes the navigation request and returns the path information.
   - This step is part of the `NavigationRequest.srv` service response.
5. CI agent receives the path and escorts the visitor agent to the required room.
   - The CI agent uses the path information provided by the BI agent to complete the escort.
   - This final escort is also visualized in RViz.

5 Simulation Setup and Visualization

The system is simulated and visualized using RViz. The map_publisher publishes the campus on the `/map` topic, which is visualized in RViz, see figure 3. The agents are spawned in the environment and their paths are visualized in RViz. Dark blue color represents the campus incharge agent, light blue color represents the visitor agent.



(a) Campus Map

(b) Map Labels

Figure 3: Overall Map

5.1 Scenarios

- **Scenario 1**: Visitor agent requests escort to the required room within the building authorized.

  Simulation Video Link - VIDEO

  In this scenario, the visitor requested to go to Building B, room 3. The campus agent escorted the visitor using a vehicle to Building B and then interacted with the building agent to retrieve the path to room 3 within the building. The campus agent then escorted the visitor to room 3.

- **Scenario 2**: Visitor agent requests escort to the required room within the building, but the building incharge agent denies the request.

  Simulation Video Link - VIDEO

  In this scenario, the visitor wanted to go to Building C room 3. The campus agent escorted the visitor to Building C. However, upon interacting with the building in-charge agent, the request for access to room 3 was denied. This may be due to restricted access. The campus agent could not proceed, and the visitor was informed about the denial.

- **Scenerio 3**: Single visitor requests for OOS.

  Simulation Video Link - VIDEO

  In this scenerio, visitor requests for OOS and then BI agent is engaged with the visitor for some amount of time till visitor leaves. During this time, the BI agent addresses any concerns or questions the visitor may have, ensuring their needs are met while managing the OOS protocol.

- **Scenerio 4**: Multiple visitor agents with single CI agent.

  Simulation Video Link - VIDEO

  In this scenerio, 3 visitors made simultaneous escort requests so CI agent first, took them to their desired buildings one by one, and then escorted them to the room. This highlights the CI agent's ability to manage multiple requests efficiently, ensuring each visitor reaches their destination.

6 Performance

The performance of the campus and building in charge agents can be evaluated based on several key metrics, reflecting their effectiveness in managing escort requests and overall visitor satisfaction. The following performance metrics have been established:

Agent Type	**Performance Metrics**	**Value**
Campus In-Charge Agent	Successful Escorts	100%
	Average Response Time	21.67 sec
Building In-Charge Agent	Request Handling Efficiency	99.9% (0.001 failure rate)
	Access Denials	1 (for unauthorized access)
	Visitor Interaction Time	33 sec (average)

Table 1: Performance Metrics for CI and BI agents

7 Conclusion

This multi-agent system efficiently facilitates campus tours by coordinating CI, BI, and Visitor agents through a well-defined communication protocol. Using ROS2 Foxy and RViz for visualization, the system enables real-time simulation and evaluation of agent performance.

8 Code

Source Code - Github