MAAS - Project Report
Commitment Issues

Sushant Vijay Chavan
Ahmed Faisal Abdelrahman
Abanoub Abdelmalak

January 17, 2019

Contents

1   Introduction

This project was completed as a part of the project work for the Multi-Agent and Agent Systems course. In this project we implemented two stages (Packaging and Delivery) of the Flying Saucers Bakery project. In addition we implemented visualization of the delivery stage as a street network with moving trucks to show the delivery in action.

TODO

2   Packaging Stage

This stage deals with completion of the post baking stages and then packaging the products into boxes so that they are ready for delivery. The stage receives the baked and cooled products from the cooling racks and preforms the final preparation of the products such as sprinkling and decorating according to the recipe for the product. Once the products are ready, they are packed into boxes based on the delivery date priority of the orders. Once all the boxes of a given product type in an order are ready, they are sent to the delivery stage for transportation.

Figure 3 gives an overview of the architecture of the packaging stage. This stage consists of three agents and in addition communicates with the CoolingRacks agent to receive the cooled products. The GenericItemProcessor takes care of all the post-baking steps, where as the packaging agent boxes the products based on the order priority. The Loading bay keeps track of the produced quantity of the products and informs the delivery stage whenever all products of a type in an order are ready.
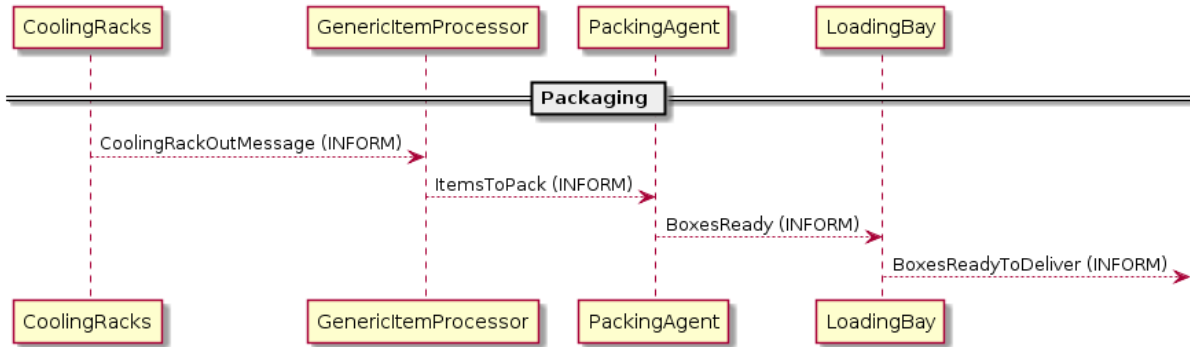

Figure 1: Packaging Stage Architecture

The output of this stage is a set of boxes that consists of a fulfilled item from a customer order. For example, if a customer order consists of 10 breads and 15 donuts, assuming that the loading bay sees that the 15 donuts of this order are available, it sends these boxes to the delivery stage. Please note that not all products of this order are yet complete and that the loading bay will wait until all the breads of this order are available before sending out a new message to the delivery stage.

2.1   Agents

1. *Generic Item Processor* (Owner: Abanoub):
   This agent is responsible for receiving the items sent by the cooling racks, prepare them for packaging and send them to the packaging agent to be packed. Based on the scenario

files used for the simulation and the bakery this agent belongs to, it creates a list with all the products. Using the scenario file, each product is acquired with the processes it needs to be ready for packaging and their duration. After preparing this list of the products with their features, the agent starts to receive products from the cooling racks and start executing the processes one by one for each of them until it is done. The products are sent to the packaging agent as soon as they are processed and all the time needed for each of the processes is finished.

2. *Packaging agent* (Owner: Sushant):
   This agent is responsible for receiving the ready products and then packaging them into boxes. The packaged boxes are then sent to the loading bay so that they can be dispatched to the customer. This agent prioritizes the packaging of the received products based on the delivery time of the orders. For example, assume we have two orders requiring 10 and 20 breads respectively and that the delivery date of the first order is earlier than that of the second. Now if we receive 20 breads from the previous stage, the packaging agents first boxes the 10 breads for the first order and sends them to the LoadingBay. It then packages the remaining 10 breads as a part of the second order. It must be noted that the packaging agent sends out the boxes as soon as they are full. It expects the loading bay to store the boxes until an all boxes of a product in an order are ready.

3. *Loading Bay* (Owner: Ahmed):
   Description and message description

3 Delivery Stage

This stage handles the delivery of the packaged boxes from the bakery to the customers. The inputs to this stage are a set of boxes from the packaging stage as described in the section 2. This stage waits until all the products of an order are ready for dispatch and then requests the trucks for transportation of these boxes. Once the orders have been delivered to the customer, the trucks post the status of the delivery to the mailbox, which then informs all agents that are interested in this information.

This stage consists of 5 agents as shown the architecture diagram in figure 2. In addition this stage talks with two other agents: *LoadingBayAgent* and the *GraphVisualizationAgent*. The figure gives an overview of the communication happening between the agents in order to complete an order received from the LoadingBayAgent. The visualization of the delivery stage will be explained in detail in section 3.2. However it is helpful to know that the only agents that need to communicate with the GraphVisualizationAgent are the StreetNetworkAgent and the TruckAgents.
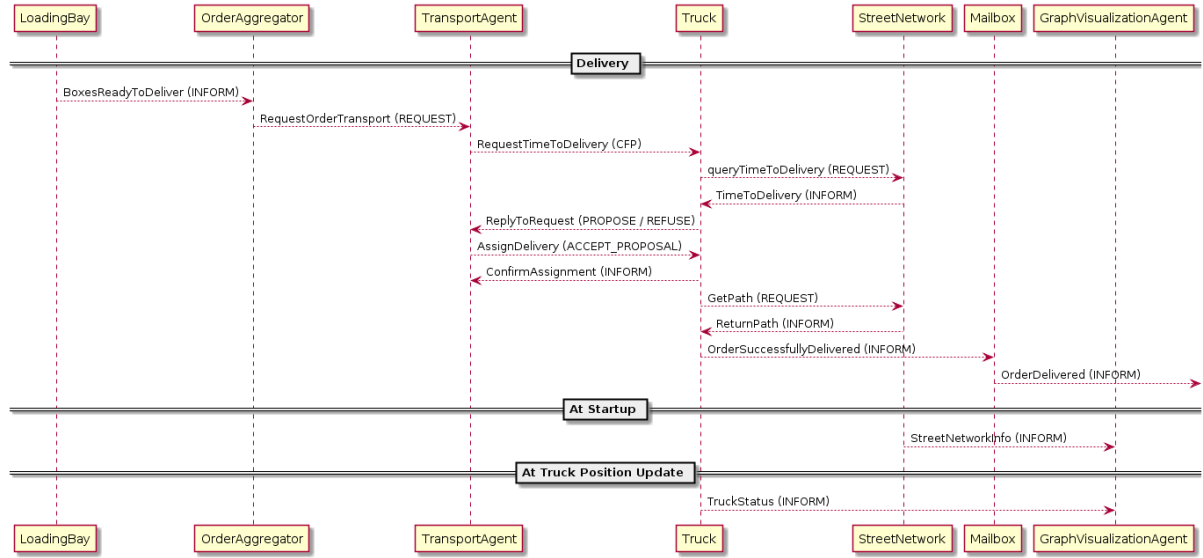

Figure 2: Delivery Stage Architecture

In order to find the best truck that can deliver the orders, the transport agent of the bakery requests for proposals from the trucks to deliver the order. Each Truck which is capable of delivery this order then queries the StreetNetwork to find the time needed by it to complete its current task and then return to the bakery to collect the products and then move towards the customer. These estimates are then sent back to the transport agent, which then chooses that truck that has the least time to delivery. Section 3.1 provides a detailed description of all the agents that are a part of the delivery stage.

3.1 Agents

1. *OrderAggregator* (Owner: Abanoub):
   Based on the idea that the loading bay will send the boxes of products that are complete, this agent is responsible for gathering all the needed information about the boxes sent by the loading bay and forward the complete orders to the transport agent after it makes

sure that the order is fulfilled. The agent works as follows: it keeps track of all the orders that the order processor broadcast. Then it waits until the loading bay sends boxes to be delivered. The boxes with their order id are used to check the order fulfillment and then send the full order to the transport agent.

2. *TransportAgent* (Owner: Abanoub):
   This agent is responsible for receiving the boxes of fulfilled orders from the order aggregator, getting the location of the bakeries and the customers of this specific order, assigning the order to the trucks with the best delivery time and handing the orders to the trucks when they arrive at the bakeries. This is achieved as follows: It keeps updating a list of customers ids associated with their orders ids sent by the order processor. Then the cycle starts when it receives boxes from the order aggregator. It uses the details sent by the order aggregator to find the location of the customer and the bakery associated with this order. After that it communicates with all the trucks working under the the same transport company to call for proposals with the time each of them needs to deliver the order then chooses the truck with the best time and assigns the order to it. When the truck arrives at the bakery to pick the order it gives the order to it.
   Each transport agent and group of trucks are associated with one transport company.

3. *TruckAgent* (Owner: Sushant):
   This agent is responsible for collecting orders from a bakery and delivering them to the customers. Each truck is associated with a transport company and multiple transport companies could exist. It is responsible for querying the estimated delivery time for an order and proposing this to the transport agent. If the proposal is accepted, it is also responsible for moving towards the bakery to collect the orders and the move towards the customers to deliver them. Finally it also informs the status of the delivery (such as the time of delivery, number of boxes) to the mailbox. Due to its highly dynamic nature, this agent has an extensive communication with other agents, as can be seen from the figure 2. It must be noted that with the current implementation, every truck can accept atmost one extra order in addition to the one it is currently handling.

   Additionally this agent also communicates with the GraphVisualizationAgent in order to send its position and status updates for visualization.

4. *StreetNetworkAgent* (Owner: Ahmed):
   Description

5. *Mailbox* (Owner: Ahmed):
   Description and message description

3.2 Graph Visualization Agent

Screenshot of the graph along with short description. Add reference to the codes source. Owners: All
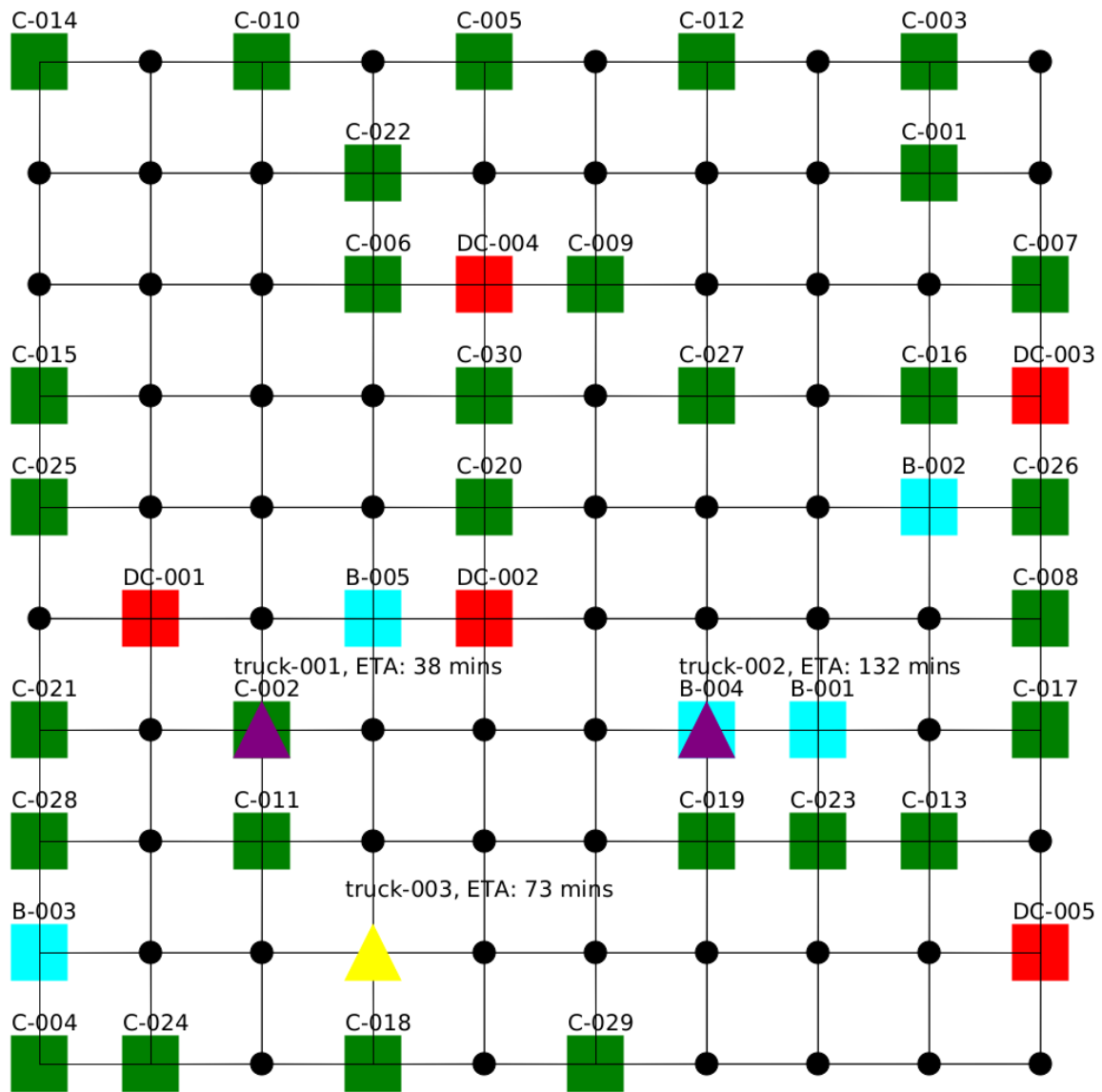

Figure 3: Visualization Screenshot

4 Entry Agent

Owner: Sushant

In order to test the two developed stages independent of the other stages, we need an agent that can trigger the first agent of the cooling racks agent to start the simulation. The EntryAgents fulfills this requirement. Using this agent, it is possible to simulate that the products from of an order have been baked and are ready for cooling, which then starts the two following stages until the order has been delivered to the customer. It should be noted that the EntryAgent simply uses the order information available in the scenario files and does not generate any data on its own.

The Entry agent has two functionalities. First, it reads the order details from the currently active scenario directory and then generates dummy messages that inform the cooling rack that the products of an order are ready. These messages are then sent to the cooling rack at the delivery time mentioned in the scenario file. Therefore the first part of this agent acts as the trigger for the two stages. Since the order processor agent was also not available on the upstream repository, we added the order processor functionality as the second functionality of the entry agent. Whenever an order is sent to the cooling rack, a corresponding order information is also broadcasted to all agents. We used the same message format as the actual order processor. This way we ensure that all our agents are compatible with the actual order processor, whenever it is available.

5 Usage Instructions

5.1 Running the two stages

5.2 Running on distributed systems

6 Conclusion