Mini UPS

Product Differentiation

Team:

Xiaodong Liu xl346
Xiao Han xh114

Front-End:

1. **Multiple information sign-up**
   Besides only signing up with amazon related user name, we are allowed to register with an email address, which will receive the email notification later.
2. **Email Notification**
   We are able to send the email to the client, as they require to change the address of a package.
3. **Full detailed order page**
   After the user logs in to the website, it will show all the package belonging to the user name with full details. Users can click the “Refresh” button or “Search” button to keep track of the newest information on all packages.
4. **Advanced Search feature**
   Client can search through all owned packages by either product name or package status or both; then it will list all the package which has this product in it, and offering a full detailed order information with the target package ID, package status, product details (including product name, product count and product description) destination and truck ID.
5. **Additional Secure Check**
   To avoid malicious users from checking the package’s information that does not belong to them. We are not only replying on Amazon to send the correct username but also double checked on our end. We will first store the username sent by amazon, but only the authorized user whose name equals the user name already exist can check the package.

Back-End:

1. **Acks Delete Policy (Hash table as container)**
   In order to keep track of the communication process with the world simulator and deal with the unexpected error message received from the world, we implemented a

policy that defined the rule of how and when to store then ack and message we send, and when to when to delete.

- We create three containers. Each container is a hash table. The key of every entry is the ack, the value is the message itself in byte.
- The first container, called commandHash, which will store all the messages we send out to the world simulator. If we receive an ack back, we will move the key-value pair of this ack to the second container.
- The second container, called waitToDelete, will hold all the kay-values in this map for 2 seconds and move them all to the next container.
- The third container, called ToDelete, will hold all the kay-value pair for another 3 seconds, and clear the hash table.

Because the second container will receive the new key-value pair at different time, therefore, 2nd and 3rd container working together will ensure the message wait at least 3 seconds, at most 5 second to delete. Furthermore, this policy provides a place to store the ack and commands, which provide a way to deal with unexpected errors when interacting with the world simulator, such as loss messages or error handling.

2. Additional Error Handler

- Pickup: it may be a chance to get an error return when we send “Ugo Pickup” command to the world simulator based on our design. Because of two reasons:
  - When Amazon asks for a truck, we will find an available truck (from our database) and send the truck id with the warehouse id to the world simulator. But before we update the truck status, there may be another ask truck command received from Amazon. Therefore, it may find the same truck.
  - To keep track of the truck's status, we have a function that queries the truck status every second from the world simulator. Even though we successfully updated the truck status based on the previous logic on time, there is still a chance the querying response overwrites the truck status.

  Therefore, we implemented a pickup error handler to handle the error message we may receive based on the previous reasons. Which will first read the original sequence number from the error message, then find the message we sent before in three containers (as mentioned in #1), read the message, if it is for pick up, extract the basic info: warehouse ID, and call the Ugo Pickup to wrap message function again, and update the database (we stored the original sequence number as one of the package information so that we can update the correct package).
- GoDeliver: To handle the error message of goDeliver, we will re-send the message we send before with a new sequence number. The original message will be found in the three hash-map containers.

3. **Process track log file**
    - We create different threads for different interactions: world simulator, amazon and front end. In order to keep track of the process, and easy debug, we create the log file for different handler. You can find it under /logs directory.

4. **Truck – Package related**
    - In order to fulfill Amazon's request, we created a truck-package relationship. This relation may change based on the status of the packages and trucks;
    - When a package is first created, we will keep storing the original sequence number received from Amazon and truck id. Therefore, we could check which package belongs to which truck at one moment.
    - When we update the pack into another status, since Amazon and World send the information per id instead of per truck, we have to use several filters to check the truck-package relation. We have built several helper functions for the database to help check it out.