

**ECE 568 ERSS: Final Project**  
**Mini-Amazon / Mini-UPS**

**Protocol Document**

**Interoperability Group 7**

Group members: Yuchen Yang(yy227), Liwen Deng(ld190), Ziqi Pei(zp33),  
Yanjia Zhao(yz476), Jiaran Zhou(jz270), Haohong Zhao(hz147), Yifan  
Li(yl506), Kaiwen Wang(kw284)

**Language and Framework:**

Our group members SHOULD use **C++** and **Python** to develop their applications. All the related frameworks and libraries such as Django, Boost, and Poco SHOULD be used to facilitate the development process. Django is a high-level Python Web framework that encourages rapid development and clean, pragmatic design. Based on our previous experience, we choose Django as our framework and **PostgreSQL** as our database.

## Interface:

Each Amazon and UPS group MUST use Google protocol buffers.

Groups will define message formats in a .proto file, use the protocol buffer compiler and use the Python protocol buffer API to write and read messages.

## Possible Interaction Scenarios:

- Amazon & UPS: **One side UPS** initializes the world and **another side Amazon** connects to the world.
- **Amazon: receive an order, connect to a warehouse and see if there is enough inventory.**
- Amazon -> UPS: Send `request_get_truck` with package ID (`shipid`), warehouse ID (`whid`), **warehouse address** (`wh_x`, `wh_y`), the **delivery destination** (`destination_x`, `destination_y`) and an optional UPS account (`ups_account`) for package pickup.
- **Amazon -> UPS: pack the order and send packed when completed.**
- UPS -> Amazon: Prepare and send `response_truck_arrived` containing trucks ID (`truck_id`) and warehouse address (`wh_x`, `wh_y`) after received `UFinished` from the world.
- Amazon -> UPS: **Amazon sends "load truck" to world simulator and sends `request_init_delivery` to UPS containing `package_id`, `delivery_location`.**
- UPS -> Amazon: Deliver **all the loaded packages the packages that Amazon requests. After delivering, send `response_delivered` to Amazon when a package is delivered and send `completions` when all deliveries are finished.**
- **Amazon -> UPS: get tracking information by specifying a `shipid`.**  
UPS -> Amazon: return tracking information containing the delivery status and location information.

## Protocol Definitions:

- Each communication message MUST have a unique sequence number and SHOULD be acknowledged by the receiver end. Otherwise, the sender SHOULD continuously send the message until receiving the corresponding ack. A timeout mechanism MAY also be applied.
- When an order is placed, Amazon SHOULD send a `whid request_get_truck` to UPS. Then UPS SHOULD send `UGoPickup` message to World.
- When UPS receive `UFinished`, it SHOULD send a `truckid, package_id and warehouse location` to Amazon. Then if Amazon receives this message and has received `APacked` message from World, it SHOULD send `APutOnTruck` message to the world.
- When Amazon receives `ALoaded` message from the world, it SHOULD send `truck ID, ship ID and destination` to UPS `to start delivering`. Then UPS SHOULD send `GoDeliver` to World.
- When UPS receive `UDeliveryMade` message from the world, it SHOULD send a confirmation message (`response_destination_changed`) to Amazon to present the completion of this order.
- Amazon MAY send a request to UPS to track the status of order using ship ID. UPS MUST store the status of every order.
- Every `shipid` MUST MAY bind with a UPS account. Users can log in their UPS accounts to see the details of all the packages.

#### Amazon -> UPS:

// this message is sent from Amazon to UPS when there is a new buy order. It can be parallel with the `APack` message.

```
Message OrderATruck{
    Required int64 whid = 1;
    Required int64 seqnum = 2;
}
```

// this message is sent from Amazon to UPS when `ALoaded` message is received.

```
Message PackageLoaded{
    Required int64 shipid = 1;
    Required int64 x = 2;
    Required int64 y = 3;
    Required int64 truckid = 4;
    Required int64 seqnum = 5;
}
```

// UPS needs to bind a `package_id` with a `truck_id`

```

message request_get_truck {
    required int64 order_id = 1;
    optional string ups_account = 2;
    required int32 warehouse_id = 3;
    required int32 location_x = 4; // warehouse location_x
    required int32 location_y = 5; // warehouse location_y
    required int32 destination_x = 6; // delivery location_x
    required int32 destination_y = 7; // delivery location_y
}

// Since truck_id is already bound to a package_id, this is
// enough.
// loaded and start delivering
message request_init_delivery {
    required int64 package_id = 1;
}

message request_change_destination {
    required int64 package_id = 1;
    required int32 new_destination_x = 2;
    required int32 new_destination_y = 3;
}

message AUCommands {
    optional request_get_truck get_truck = 1;
    optional request_init_delivery init_delivery = 2;
    optional request_change_destination change_destination = 3;
    optional bool disconnect = 4;
}

```

### UPS -> Amazon:

// this message is sent from UPS to Amazon when a truck has arrived at a warehouse.

```

Message TruckArrived{
    Required int64 truckid = 1;
    Required int64 seqnum = 2;
}

```

// this message is sent from UPS to Amazon when a package is delivered to its destination.

```

Message PackageDelivered{
    Required int64 shipid = 1;
}

```

```

        Required int64 seqnum = 2;
    }

    // Send the response to Amazon that truck has arrived at the
    warehouse
    // Change truck status as well.

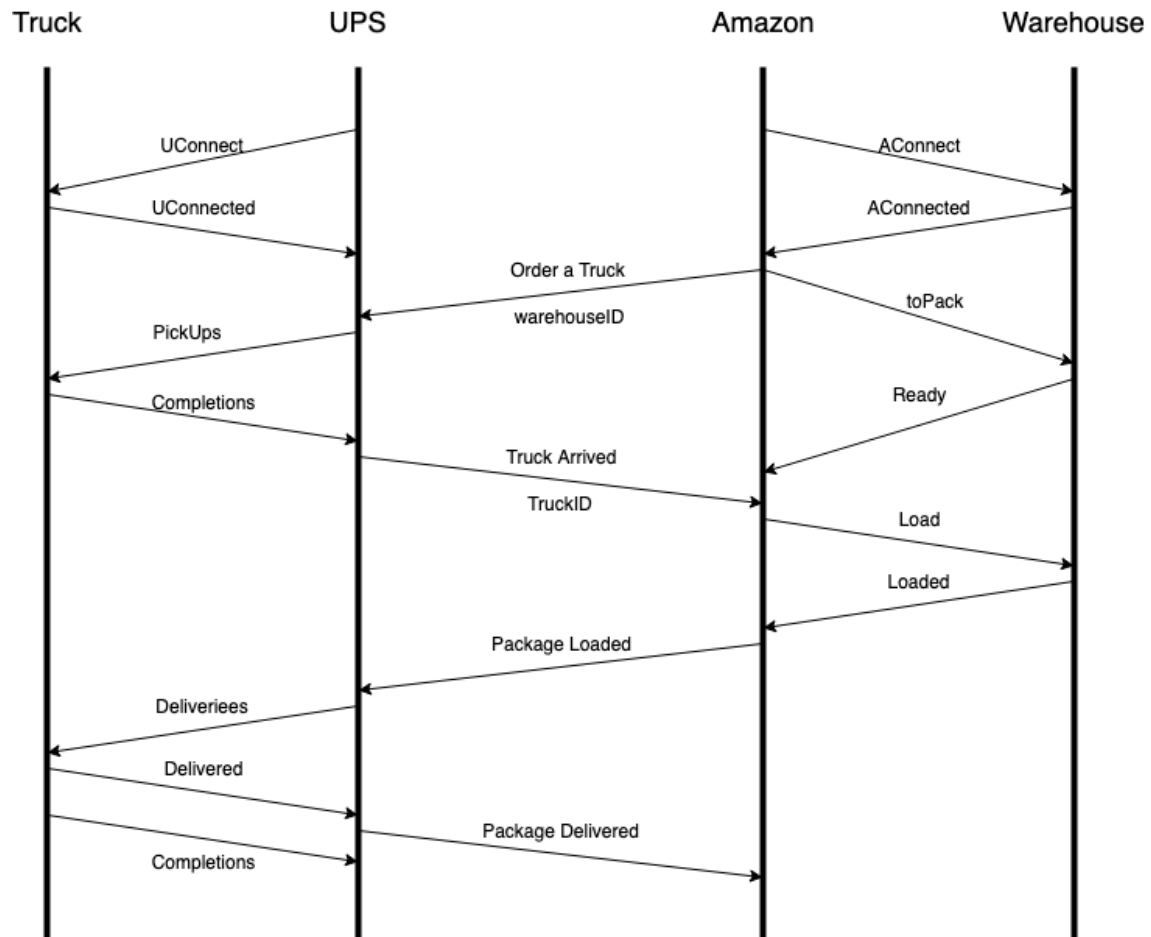
    message response_truck_arrived {
        required int32 wh_x = 1;
        required int32 wh_y = 2;
        required int32 truck_id = 3;
        required int64 package_id = 4;
    }

    //package has been delivered
    message response_package_delivered {
        required int64 package_id = 1;
    }

    //whether the destination is changed successfully
    message response_destination_changed {
        required int32 new_destination_x = 1;
        required int32 new_destination_y = 2;
        required int64 package_id = 3;
        required bool success = 4;
    }

    message UACommands {
        optional response_truck_arrived truck_arrived = 1;
        optional response_package_delivered package_delivered = 2;
        optional response_destination_changed destination_changed =
3;
        optional bool disconnect = 4;
        optional int64 world_id = 5;
    }

```



All communications between UPS and Amazon are bydirectional, including the acknowledgement from the receiver.