ECE 568 ERSS: Final Project Mini-Amazon / Mini-UPS
Protocol Document

Group Information: IG 3, Group 1.9-1.13

Zhicheng Jiang (zj78) Jingzhi Zhao (jz422)
Yuan Cao(yc541) Zixuan Chen (zc193)
Jiawei Liu (jl1188) Yikai Liu (yl906)
Yichen Huang (yh348) Suo Chen (sc831)
Ziye Xie (zx106) Zhou Fang (zf72)

Protocol Definition (using RFC terminology)

● "UA" stands for from UPS to Amazon, "AU" stands for from Amazon to UPS
The protocol added two composite messages that everything should go into:
UPS MUST send UAcommands to Amazon and Amazon MUST send AUcommands to UPS.
● UAcommands: Contains a list of truck arrivals, deliveries, change destination responses, bind UPS responses, error messages, an optional disconnect flag, and acknowledgments.
● AUcommands: Contains a list of pickup requests, delivery requests, change destination requests, bind UPS requests, error messages, an optional disconnect flag, and acknowledgments.
● Each communication message **MUST** have a unique sequence number and 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.
● UPS side **MUST** communicate with world to init a world with a specific worldID, then **MUST** send a UAinitWorld to Amazon side. Amazon **MUST** connect to that exact worldID and send back a AUconnectedWorld response.
● Amazon MUST send AUbindUPS to UPS with upsID and UPS MUST send UAbindUPSResponse with success or fail to Amazon side to indicate whether successfully bind upsID.
● After Amazon has finished packing and ready to load, it **MUST** send a AUreqPickup request to UPS. UPS **MUST** respond with a UApickupConfirmed response to confirm it has received the request.
● After a UPS truck arrives at a specified warehouse location, it **MUST** send a UAtruckArrived message to Amazon, indicating that Amazon can start loading. Amazon **MUST** respond with AUarriveConfirmed.
● After Amazon finish loading, it **MUST** send a AUreqDelivery to UPS, telling it to start deliver to the destination. UPS **MUST** respond with a UAstartDelivery.
● When UPS successfully delivered to destination **MUST** send a UAdelivered to Amazon, telling it that the package has been delivered. Amazon **MUST** respond with AUdeliverConfirmed.
● Amazon **MAY** send a query to UPS to track the status of order by AUquery. After receiving the message, UPS **MUST** respond with UAstatus.

● Amazon **MAY** send a request of AUchangeDestn if they want to change the delivery destination. UPS **MUST** respond with a UAchangeResp as a response. The response will indicate whether the change is successful.

Protocol Message

```
syntax = "proto2";
message AProduct{
  required int64 id = 1;
  required string description = 2;
  required int32 count = 3;
}

message UAinitWorld{
        required int64 worldID = 1;
        required int64 seqNum = 2;

}

message AUconnectedWorld{
        required bool success = 1;
        repeated int64 acks = 2;
}

message AUbindUPS {
        required int32 ownerID = 1;
        required int32 upsID = 2;
        required int64 seqNum = 3; // ack = success (exist), Err = fail (not exist)
}

message UAbindUPSResponse {
  required bool status = 1;
  required int32 ownerID = 2;
  required int32 upsID = 3;
  required int64 seqNum = 4; // ack = success (exist),
}
```

```
message AUreqPickup {
        required int32 whX = 1;
        required int32 whY = 2;
        required int32 whID = 1;
        required int32 destinationX = 23;
        required int32 destinationY = 34;
        required int64 shipID = 4;  //Amazon generate shipID
        optional int32 upsID = 5  //Amazon check whether bind exist. Otherwise don't add upsID
        repeated AProduct products = 6;
        required int64 seqNum = 7;

}

message UApickupConfirmed {
        required int32 whX = 1;
        required int32 whY = 2;
        required int32 whID = 1;
        required int64 shipID = 23;
        required int32 truckID =  34;
        requiredrepeated int64 seqNumacks = 45;
}

message UAtruckArrived {
        required int32 whX = 1;
        required int32 whY = 2;
        required int32 whID = 1;
        required int64 shipID = 23;
        required int32 truckID = 34;
        required int64 seqNum = 45;
}

message AUarriveConfirmed {
        required int32 whX = 1;
        required int32 whY = 2;
        required int64 shipID = 3;
        required int32 truckID = 4;
        requiredrepeated int64 seqNumacks = 5;
}

message AUreqDelivery {
        required int64 shipID = 1;
        required int64 seqNum = 2;
}
message UAstartDelivery {
```

```
        requiredrepeated int64 seqNumacks = 1;
}

message AUquery {
        required int64 shipID = 1;
        required int64 seqNum = 2;
}

message UAstatus {
        required string status = 1;
        required int64 seqNumacks = 2;
}

message UAdelivered {
        required int64 shipID = 1;
        required int64 seqNum = 2;
}

mesage AUdeliverConfirmed {
        required int64 shipID = 1;
        required int64 seqNumacks = 2;
}

message AUchangeDestn {
        required int64 shipID = 1;
        required int32 destinationX = 2;
        required int32 destinationY = 3;
        required int64 seqNum = 4;
}

message UAchangeResp {
        required bool success = 1;
        required int64 seqNumacks = 2;
}

message Err{
        required string err = 1;
        required int64 originseqnum = 2;
        required int64 seqnum = 3;
}

message UAcommands{
        repeated UAtruckArrived truckArr = 1;
        repeated UAstatus status = 2;
```

```
        repeated UAdelivered delivered = 2;
        repeated UAchangeResp changeResp = 3;
        repeated UAbindUPSResponse bindUPSResponse = 4;
        repeated Err err = 6;
        optional bool disconnect = 7;
        repeated int64 acks = 8;
}

message AUcommands{
        repeated AUreqPickup pickup = 1;
        repeated AUreqDelivery delivery =  2;
        repeated AUquery query = 3;
        repeated AUchangeDestn changeDest = 3;
        repeated AUbindUPS bindUPS= 4;
        repeated Err err = 5;
        optional bool disconnect = 6;
        repeated int64 acks = 7;
}
```