ECE 568 Final Project Protocol Writeup

IG 5

Rui Yang ry70, **Tongbo Liu** tl259, **Yingcong Fang** yf110, **Qingyang Xu** qx37, **Xuan Yu** xy93,
Haisong Mei hm171, **Yijun Mao** ym134, **Chixiang Zhang** cz130

04/08/2020

For all the communications between Amazon and UPS, the receiver side SHOULD send an ACK. If the sender side does not receive the ACK within a given time, it MUST resend the message. For communications between Amazon and the world simulator, as well as UPS and the world simulator, both sides should send an ACK when receiving the message from the other side. For example, when the Amazon side sends the world simulator an APack request. The world simulator would first respond with an ACK. Then it sends Amazon an APacked response. The Amazon side should finally respond with an ACK to finish this communication. We mention that here but not in the sections below to avoid the redundancy in descriptions.

1 Introduction

This writeup describes the workflow of mini Amazon and mini UPS and the interaction between these two parts under the framework of ECE 568 final project. The writeup consists of four parts: overview, Amazon workflow, UPS workflow, and definitions of messages between Amazon and UPS following the Google Protocol Buffer.

The overview part demonstrates the general workflow of both parts (Amazon and UPS) and interactions between them.

The Amazon workflow part goes through the entire process of the Amazon side starting from connecting to the world until the package gets delivered.

Similarly, the UPS workflow part lists the steps involved in its entire process.

The last part gives definitions of messages transferred between Amazon and UPS using the Google Protocol Buffer format.

We acknowledge there would probably still be modifications of the entire process as well as details. This is a premature writeup and is subject to changes in the future.

2 Overview

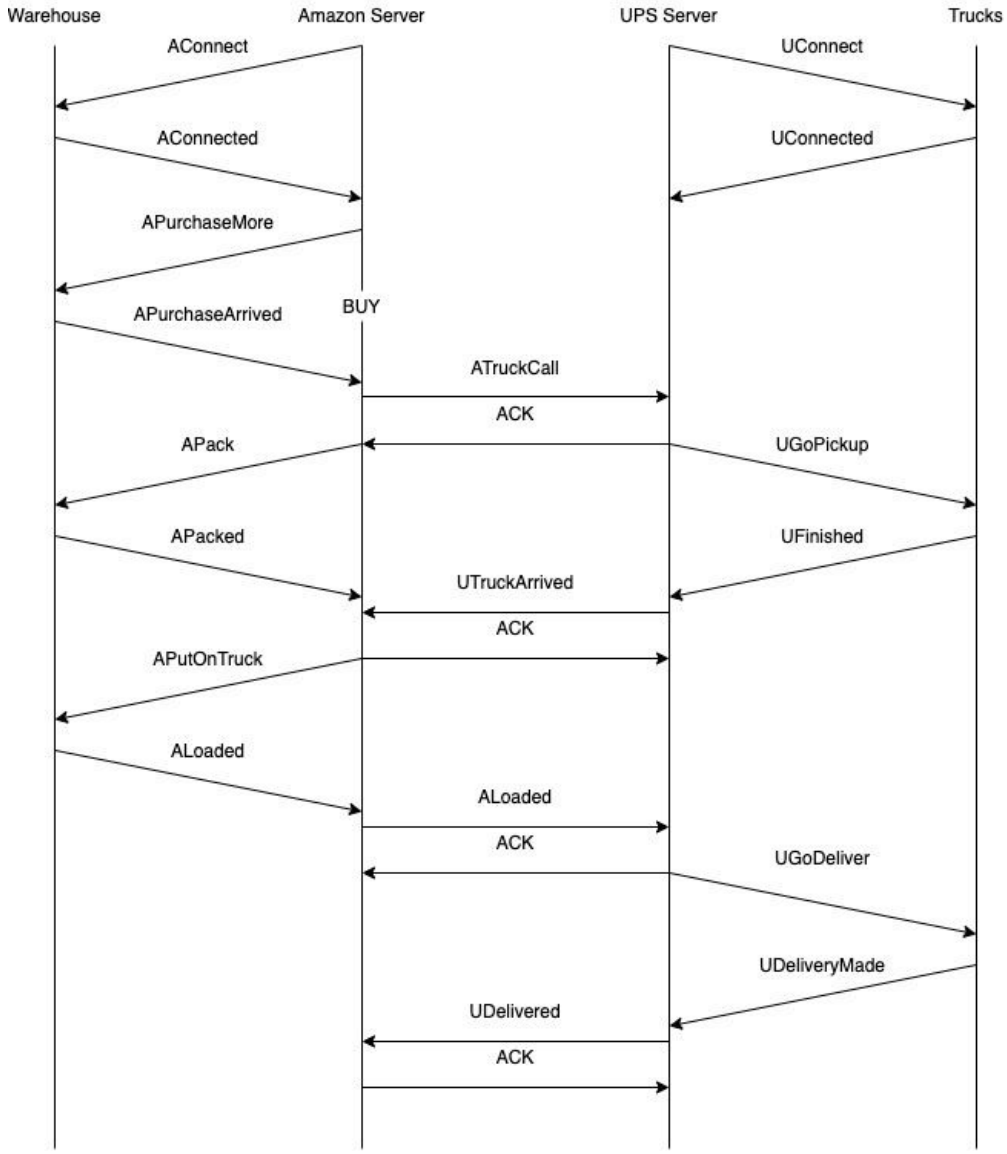

Figure 1: Project workflow

The entire workflow, which is illustrated by the figure above, could be summarized for both sides as following:

Amazon side: connect to the world -> buy products -> call for a truck from UPS -> pack the products -> make sure truck has arrived at warehouse -> load the package -> inform UPS the package has been loaded -> get informed from UPS when package gets delivered

UPS side: connect to the world -> get notified from Amazon to send a truck -> send a truck to pick up packages -> get informed from Amazon when loading finished -> delivery -> inform Amazon when delivery finished

3 Amazon Workflow

Step 1: CONNECT
Amazon MUST create a world or Connect to an existing world.

Request
Case 1: Create a new world

```
message AConnect{
  optional int64 worldid = 1;
  repeated AInitWarehouse initwh = 2;
  required bool isAmazon = 3;
}

message AInitWarehouse{
  required int32 id = 1;
  required int32 x = 2;
  required int32 y = 3;
}
```

Case 2: connect to an existing world
The message is basically the same as that of creating a new world. The only difference is the world id is left blank when creating a new world and must take a value when connecting to an existing world.

Response

```
message AConnected{
  required int64 worldid= 1;
  required string result = 2;
}
```

Step 2: BUY

The customer buys a product on the website, and wants to deliver it to some address. After the order is placed, the Amazon side MUST check the warehouse database to see if there is still enough storage of that item. If yes, update the warehouse database directly, and create an order. Otherwise, the Amazon side MUST send a APurchaseMore message to the world simulator. The Amazon side SHOULD receive an ACK from the world simulator. If Amazon does not receive the response within a given time, it MUST resend the message. After receiving the ACK from the world simulator, the Amazon side updates the warehouse database.

Request

```
message APurchaseMore{
  required int32 whnum = 1;
  repeated AProduct things = 2;
  required int64 seqnum = 3;
}

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

Response

```
message APurchaseArrived{
  required int32 whnum = 1;
  required int64 seqnum = 2;
}
```

Step 3: communicate with UPS to call for a truck

Amazon side MUST send UPS side a message to call for a truck if there's no truck in the warehouse. If there's already a truck, Amazon side MAY not send a new request. UPS side SHOULD response with an acknowledgement. When the Amazon side does not receive the acknowledgement from UPS within a given time, it MUST resend the request.

Request

```
message ATruckCall {
  required int64 package_id = 1;
  required int32 whnum = 2;
  required int32 dest_x = 5;
  required int32 dest_y = 6;
  required int64 seqnum = 7;
}
```

Step 4: pack

Amazon MUST check the warehouse database for the storage of products of the order. If there's sufficient inventory, it MUST send the world simulator. It SHOULD receive an ACK from the world simulator and then update the warehouse database. If Amazon does not receive response from the world simulator within a given time, it MUST resend the request. If there's not sufficient inventory for the product, it MUST send the world simulator an APurchaseMore message to buy the products.

Request

```
message APack{
  required int32 whnum = 1;
  repeated AProduct things = 2;
  required int64 shipid = 3;
  required int64 seqnum = 4;
}

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

Response

```
message APacked {
  required int64 shipid = 1;
  required int64 seqnum = 2;
}
```

Step 5: make sure the truck has arrived

The UPS side MUST send Amazon a message when the truck arrives at the warehouse. The Amazon side SHOULD respond with acknowledgement. If the UPS side does not receive the response within a given time, it MUST resend the message.

Request

```
UTruckArrived {
  required int64 package_id = 1;
  required int32 whnum = 2;
  required int32 truck_id = 3;
  required int64 seq_num = 4;
```

}

Step 6: load

Amazon loads a shipment on to a truck. Amazon side MUST have received APacked and UTruckArrived.

Request

```
message APutOnTruck{
  required int32 whnum = 1;
  required int32 truckid = 2;
  required int64 shipid = 3;
  required int64 seqnum = 4;
}
```

Response

```
message ALoaded{
  required int64 shipid = 1;
  required int64 seqnum = 2;
}
```

Step 7: inform UPS the package is ready for delivery

After loading finished, the Amazon side MUST send the UPS side an ALoaded message. It should receive ACK from the UPS side. If the Amazon side does not receive the ACK within a given time, the Amazon side MUST resend the ALoaded message.

Request

```
message ALoaded {
  required int64 shipid = 1;
  required int32 truckid = 2;
  required int64 seqnum = 3;
}
```

Step 8: UPS tell the Amazon side the product has been delivered

After the product has been delivered successfully. The UPS side MUST send the Amazon side a message to tell it the product has been delivered. The Amazon side SHOULD reply with an ACK and then update the warehouse database after receiving the ACK. If the UPS side does not receive the ACK within a given time, it MUST resend the message.

Request

```
message UDelivered {
  required int64 packageid = 1;
  required int32 truckid  =2;
}
```

4 UPS Workflow

Step 1: CONNECT

MUST Create a world or Connect to an existing world.

Request

Case 1: Create a new world of several Trucks

```
message UConnect{
  optional int64 worldid = 1;
  repeated UInitTruck trucks=2;
  required bool isAmazon = 3;
}

message UInitTruck{
  required int32 id = 1;
  required int32 x=2;
  required int32 y=3;
}
```

Case 2: connect to an existing world

The only difference to creating a world is worldid should now take a value

Response

```
message UConnected{
  required int64 worldid = 1;
  required string result = 2;
}
```

Step 2: Receive request from Amazon Server

UPS side MUST receive UTruckReq from Amazon server. UPS side MUST respond with an acknowledgement of UTruckRes.

Request from Amazon

```
message ATruckCall {
  required int64 package_id = 1; no need
  required int32 whnum = 2;
  required int32 dest_x = 3; no need
  required int32 dest_y = 4; no need
  required int64 seqnum = 5;
}
```

Step 3: pickups

UPS MUST send the Truck world simulator a UGoPickup request. It SHOULD receive an ACK from the world simulator. If UPS does not receive response from the world simulator within a given time, it MUST resend the request.

Request

```
message UGoPickup{
  required int32 truckid = 1;
  required int32 whid = 2;
  required int64 seqnum = 3;
}
```

Response

```
message UFinished{
  required int32 truckid = 1;
  required int32 x = 2;
  required int32 y = 3;
  required string status = 4;
  required int64 seqnum = 5;
}
```

Step 4: make sure the truck has arrived

The UPS side MUST send Amazon a UTruckArrived message when the truck arrives at the warehouse. The Amazon side MUST respond with acknowledgement of UTruckReady. If the UPS side does not receive the response within a given time, it MUST resend the message.

Request to Amazon

```
message UTruckArrived {
  required int64 package_id = 1;
```

```
  required int32 whnum = 2;
  required int32 truck_id = 3;
  required int64 seq_num = 4;
}
```

Step 5: deliveries

UPS MUST send the Truck world simulator UGoDeliver request. It SHOULD receive an ACK from the world simulator. If one of the deliveries is done, UDeliveryMade SHOULD be received. If all of the deliveries are done, UFinished SHOULD be received. If UPS does not receive response from the world simulator within a given time, it MUST resend the request.

Request

```
message UDeliveryLocation{
  required int64 packageid = 1;
  required int32 x = 2;
  required int32 y = 3;
}

message UGoDeliver{
  required int32 truckid = 1;
  repeated UDeliveryLocation packages = 2;
  required int64 seqnum = 3;
}
```

Response

```
One of the delivery is done
message UDeliveryMade{
  required int32 truckid = 1;
  required int64 packageid = 2;
  required int64 seqnum = 3;
}

All of the deliveries are done
message UFinished{
  required int32 truckid = 1;
  required int32 x = 2;
  required int32 y = 3;
  required string status = 4;
  required int64 seqnum = 5;
```

}

Step 6: UPS tell the Amazon side the product has been delivered

After the product has been delivered successfully. The UPS side MUST send the Amazon side a message to tell it the product has been delivered. The Amazon side SHOULD reply with an ACK and it SHOULD update the warehouse database after receiving the ACK. If the UPS side does not receive the ACK within a given time, it MUST resend the message.

Request

```
message UDelivered {
  required int64 packageid = 1;
  required int32 truckid  =2;
}
```
[1]

5 Amazon-UPS Communication Protocol

New Protocol

- Distinct ACK and Seqnum, created from Amazon side
- Caution on primitive type. Otherwise ack, seqnum not unique
- UA_GoDeliver replaces UA_Loaded

```
# Using

message UA_Connect {
  required int64 worldid =  1;
}
# Add: ups side need product info
message UA_TruckCall {
  required int64 package_id = 1;
  repeated AProduct products = 2;
  required int32 whnum = 3;
  required int64 owner_id =4;
  required int32 dest_x = 5;
  required int32 dest_y = 6;
  required int64 seqnum = 7;
}
```

1

```
# changed
message UA_GoDeliver {
    required int32 truckid  = 1;
  required int64 packageid = 2;
  required int32 x = 3;
  required int32 y = 4;
  required int64 seqnum = 5;
}

message UA_TruckArrived {
  required int32 whnum = 1;
  required int32 truck_id = 2;
  required int64 seqnum = 3;
}

message UA_Delivered {
  required int64 packageid = 1;
  required int32 truckid  = 2;
  required int64 seqnum = 3;
}


message UA_Commands{
  repeated UA_TruckCall truckCall = 1;
  repeated UA_GoDeliver goDeliver = 2;
  repeated int64 acks = 3;
}

message UA_Responses{
  repeated UA_TruckArrived truckArrived = 1;
  repeated UA_Delivered delivered = 2;
  repeated int64 acks = 3;
}
```

Not using

```
message ATruckCall {
  required int64 int64 package_id = 1;
  required int32 whnum = 2;
  required int32 dest_x = 5;
  required int32 dest_y = 6;
  required int64 seqnum = 7;
}

message UTruckArrived {
  required int64 package_id = 1;
  required int32 whnum = 2;
  required int32 truck_id = 3;
  required int64 seq_num = 4;
}

message ALoaded {
  required int64 shipid = 1;
  required int32 truckid = 2;
  required int64 seqnum = 3;
}


message UDelivered {
  required int64 packageid = 1;
  required int32 truckid  =2;
}

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

message AUMessages {
  repeated ATruckCall call = 1;
  repeated UTruckArrived arrived = 2;
  repeated ALoaded loaded = 3;
  repeated UDelivered = 4;
  repeated AUErr = 5;
  repeated int64 acks = 6;
}
```