Interoperability Group C:

Protocol Specification

Engineering Robust Server Software Final Project

Authors: Arda Icmez (ai67), Connor Grehlinger (cmg88),Yunzhen Zou(yz392), Liang Zhang (lz139), Zhimin Chang(zc86), Jichun Shen(js751), Yan Su(ys206), Wei Zhang(wz88)

Web Interface Technologies:

~~Amazon and UPS MUST communicate over HTTP with JSON. This is a common and well-documented technique of interacting over the web. JSON provides an easy way to transmit text over client-server connections representing objects, data and requests. An example of a JSON representation of a something like a track package query illustrates this well (see JSON Example on last page).~~

Amazon and UPS MUST communicate using Google protocol buffers. See end of Protocol specification document for revisions.

Languages and Frameworks for Development:

Project groups within this interoperability group MAY develop server applications from languages like Python, C++ and Java. Project groups may make use of existing frameworks for these languages such as: Django, Boost, Poco, and Spring. Though different languages and frameworks MAY be used, each group MUST be able to communicate with each other with a defined interface allowing for all functionality requirements ~~(i.e. JSON over HTTP)~~ (i.e. Google protocol buffers).

Possible Amazon UPS Interaction Scenarios:

This is simply a list of scenarios which may occur during the operation of the Amazon and UPS servers.

A->U: cancel shipment(order_id, product_id, wearhouse_id)
A->U: pick up order(warehouse_id, order_id, product_id)
A->U: get tracking info(order_id)
U->A: send tracking info(last_warehouse_location, delivery status(in-transit/delivered/error), time_of_last_checkpoint)
U->A: confirm delivery(order_id)
A->U: send warehouse address to UPS(warehouse_id)
U->A: send order completion message to AMAZON(order_id)

A->U: send return message to UPS to return package back to warehouse(package_id)

~~Protocols:~~

- ~~NOTIFY_DELIVERY <package_id> : Used by the UPS to notify AMZN about the delivery~~
- ~~NOTIFY_PICKUP<package_id, warehouse_id>: Used by AMAZON to send order and warehouse message to UPS~~
- ~~TRACK_PACKAGE<package_id?>: Used by AMZN to request package state~~
- ~~DELIVERY_ACK <order_id?> : Used by AMZN to acknowledge the delivery made by UPS~~
- ~~TRUCK_READY <truck_id> : Used by UPS to tell AMZN that the truck has arrived at the warehouse that is specified~~
- ~~TRUCK_DONE <truck_id> : Used by UPS to tell AMZN that the truck has finished all of its deliveries~~

Basic Principles:

Order and order information (Amazon's responsibility)

Package and shipment (UPS's responsibility)

~~An order can have one or more packages, AMZN handles the status of order by querying packages status from UPS~~

A package handled by UPS can have one or more products.

Protocol Specifications:

1. ~~Both Amazon and UPS MUST acknowledge any request received by the other party (they must acknowledge they have received the request).~~
   1.1. Each party ~~SHOULD~~ MUST respond to that request ~~following the acknowledgement~~ with information to fulfill that request.
   1.2. If the request cannot be fulfilled, information pertaining to why the request cannot be fulfilled ~~MUST~~ SHOULD be supplied in the receiving server's output ~~response.~~
2. ~~Amazon MAY make a request to cancel an order if UPS has not yet shipped that particular order.~~ Amazon MUST not make a request to cancel an order. Amazon MAY request a change of package destination if the package is not yet loaded on a truck.
   ~~2.1. UPS MUST cancel the order if it has not yet shipped~~
   ~~2.2. UPS MUST deny the cancellation request if the order has already shipped~~

~~2.3. UPS MUST inform Amazon of either outcome~~

2.4. UPS MUST inform Amazon of package status information

3. Amazon MAY make a request to track packages
   3.1. UPS SHOULD respond with package location information if it can be obtained. Else UPS MUST notify Amazon that no tracking information is available.
4. Amazon SHOULD make a request to UPS to ship ~~items~~ products contained in an order
   4.1. UPS MUST determine how those items in the order request will be shipped
5. UPS MUST notify Amazon when a delivery is completed
6. ~~(BONUS)Customers MAY want to return packages. In this case, UPS truck should be able to go to the given coordinates, pick up the package from the customer, and drop it off in a warehouse of: 1) Choice 2) A specified warehouse from Amazon~~
   ~~6.1. Customer may want to return either the whole package, or distinct items from that package.~~
7. ~~(BONUS)Customers MAY want their order sent together. In this case, UPS Truck will have to deliver some packages from a warehouse to another warehouse(s).~~

~~JSON Example:~~

~~{~~
~~"package": {~~
~~"package_id": "",~~
~~"status": "",~~
~~"trucks": {~~
~~"truck_id": "",~~
~~"location_id": "",~~
~~"packages": "",~~
~~"status": "idle" / "busy"~~
~~}~~
~~}~~
~~}~~
~~{~~
~~"warehouse" : "(x, y)"~~
~~"commodity" : {~~
~~"commodity_id" : "123"~~
~~"amount" : "100"~~
~~}~~
~~}~~
~~{~~
~~"truck": {~~

~~"truck_id": "",~~
~~"location_id": "",~~
~~"packages": "",~~
~~"status": "idle"~~
~~}~~
~~}~~

NEW Version

NOTE 1:
For Communication between UPS and Amazon, UPS acts as the Server and Amazon acts as the Client.

NOTE 2:
UPS will create a new world without using the "init_world" program provided. When the world is created, UPS will get a worldid from the world and then sends it to Amazon (The first message A received from U). So for Amazon to connect to the same world, you can include a read from stdin snippet to input the worldid.

New Mod:
In AUCommands, change finished to disconnect;
In UACommands, change disconnect to finished.

```
// A -> U
message Loaded {
        required int64 packageid = 1;
}

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

message Warehouse {
        required int32 whid = 1;
        required int32 whx = 2;
        required int32 why = 3;
}
```
****

The first Message Amazon needs to send to UPS is all the warehouse information so that UPS can store warehouse info into the database.
In this way, when the client makes a purchase, Amazon only needs to include the **whid** in the **Purchase Message**.
NOTE: Also add one new repeated field (whinfo) in the AUCommands

****

```
message Purchase {
        required int32 whid = 1;
        required int32 x = 2;
        required int32 y = 3;
        // required int32 wh_x = 4;
        // required int32 wh_y = 5;
        repeated Product things = 4;
        required int64 amazonuserid = 5;
        required int64 orderid = 5;
        optional string int64 upsuserid = 6 [default=-1];
        required bool isprime = 7;
}
```

****

upsuserid in the Purchase Message is used in the following way:

First of all, when a client buys something on Amazon, Amazon would send a AUCommands which contains a Purchase sub-message in it.

Then here comes two scenarios:
1. Before a client get an UPS account, this field is useless and has the default value.
2. After the client registered in the UPS website and get a unique **upsuserid**, then he/she may put this **upsuserid** into his/her Amazon profile (which means Amazon needs to leave a blank for the client to add this information if available). After adding the **upsuserid** in Amazon, whenever the client buy anything from then on, Amazon needs to send the Purchase Message including the **upsuserid** so that we UPS can associate the packages with the **upsuserid** on UPS side.

****

```
message ChangeDes {
        required int64 pkgid = 1;
        required int32 x = 2;
        required int32 y = 3;
}
```

****

ChangeDes is considered as a **bonus point**. Since the assignment spec requires that a client should be able to change the destination before the package goes out for delivery. While by adding this message, the client can also change the destination on Amazon side, which means after the client changes the destination, Amazon needs to send to UPS the ChangeDes Message.

****

```
// A to U Command
message AUCommands {
        optional Purchase pc = 1;
        optional Loaded ldd = 2;
        optional ChangeDes chdes = 3;
        optional bool finished = 4;
        optional bool disconnect = 4;
        repeated Warehouse whinfo = 5;
}


// U -> A
message PackageStatus {
        required int64 pkgid = 1;
        required string status = 2;
}

message DesChangeResult {
        required int64 pkgid = 1;
        required bool res = 2;
}
```

****

Bonus Point:

Corresponding to the ChangeDes Message Amazon send to UPS.

****

```
message StartLoad {
        required int32 whid = 1;
        required int32 truckid = 2;
        repeated int64 packageid = 3;
}

message PackageID {
        required int64 pkgid = 1;
        required int64 amazonuserid = 2;
        required int64 orderid = 2;
}

// U to A Command
message UACommands {
        repeated StartLoad sl = 1; //tells Amz truck has arrived at the given warehouse
        optional PackageID pid = 2;
        repeated PackageStatus pks = 3;
        optional DesChangeResult dcr = 4;
```

```
        optional int64 worldid = 5;
        optional bool disconnect = 6; // disconnect from Amazon
        optional bool finished = 6;
}
```