**syntax = "proto2";**

**delete: package internalComm;**

**message AWarehouse{**

**required int32 id = 1;**

**required int32 x = 2;**

**required int32 y = 3;**

**}**

Amazon SHOULD store all the warehouses' information after calling AInitWarehouse (A.proto) and MUST send related information while calling ASendTruck (IG.proto).

**message ASendTruck{**

**required AWarehouse whs = 1;**

**repeated APackageInfo packages = 2;**

**}**

Once one or more products has been purchased by the user, the Amazon SHOULD send APack (A.proto) to the warehouse as well as send ASendTruck (IG.proto) to the UPS. ASendTruck message MUST contains the Awarehouse (IG.proto).

**message USendWorldID{**

**required int32 worldid = 1;**

**}**

This message is used to send the worldID to the Amazon.

**message UTruckSent{**

**required int32 truckid = 1;**

**repeated APackageInfo packages = 2;**

**}**

This message is used to inform the Amazon server that we have already sent the truck to the assigned warehouse. All the packages assigned to be loaded by this truck will also be sent within this message.

**message UTruckArrived{**

**required int32 truckid = 1;**

**}**

This message is used to inform the Amazon server that the UPS-assigned truck has already arrived at the specific warehouse that was requested by the Amazon server. The Amazon server just needs the truckid and it will begin loading the packages that have already been assigned to this truck according to this truckid.

**message APackageInfo{**

**required int64 packageid = 1;**

**required int32 x = 2;**

**required int32 y = 3;**

**optional string ups\_user\_name = 4;**

**}**

**message AStartDelivery{**

**required int32 truckid = 1;**

**repeated APackageInfo packages = 2;**

**}**

When Amazon receives both APacked (A.proto) and UTruckArrived (IG.proto), Amazon SHOULD call APutOnTruck (A.proto) to load. Then Amazon MAY receives ALoaded (A.proto) from the warehouse. At this time, the Amazon SHOULD call AStartDelivery (IG.proto) to let the truck to deliver. Amazon MUST contains truckid and PackageInfo (IG.proto) in AStartDelievery.

**message UDelivered{**

**required int64 shipid = 1;**

**}**

This UDelivered message is used to inform the Amazon server that we have successfully delivered the package with this package’s ship-id(aka package-id). Given that each truck may cover many packages, we will send one delivered message when each package has been delivered.

**message UInitTruck{**

**required int32 id = 1;**

**required int32 x=2;**

**required int32 y=3;**

**}**

The UInitTruck message is used to initialize the truck with its ID and coordinates.

**message UConnect{**

**optional int64 worldid = 1;**

**repeated UInitTruck trucks=2;**

**required bool isAmazon = 3;**

**}**

**The UConnect message is used to connect an existing world or create a new world if no world id is specified. It needs to have all the trucks initialized when the new world is created as well.**

**message UConnected{**

**required int64 worldid = 1;**

**required string result = 2;**

**}**

**The UConnected message is the response of the UConnect message. It includes the world id that has been created.**

**message AtoUCommands{**

**repeated ASendTruck sendtrucks = 1;**

**repeated AStartDelivery startdelivery = 2;**

**}**

**message UtoAResponses{**

**repeated UTruckSent trucksent = 1;**

**repeated UTruckArrived arrived = 2;**

**repeated UDelivered delivered = 3;**

**optional UConnected connected = 4;**

**}**

**The UtoAResponses message is the response to the Amazon.**

**/\* Probable Message \*/**

message ATraceShip{

required int64 shipid = 1;

required int64 seqnum=2;

}

message UShipInfo{

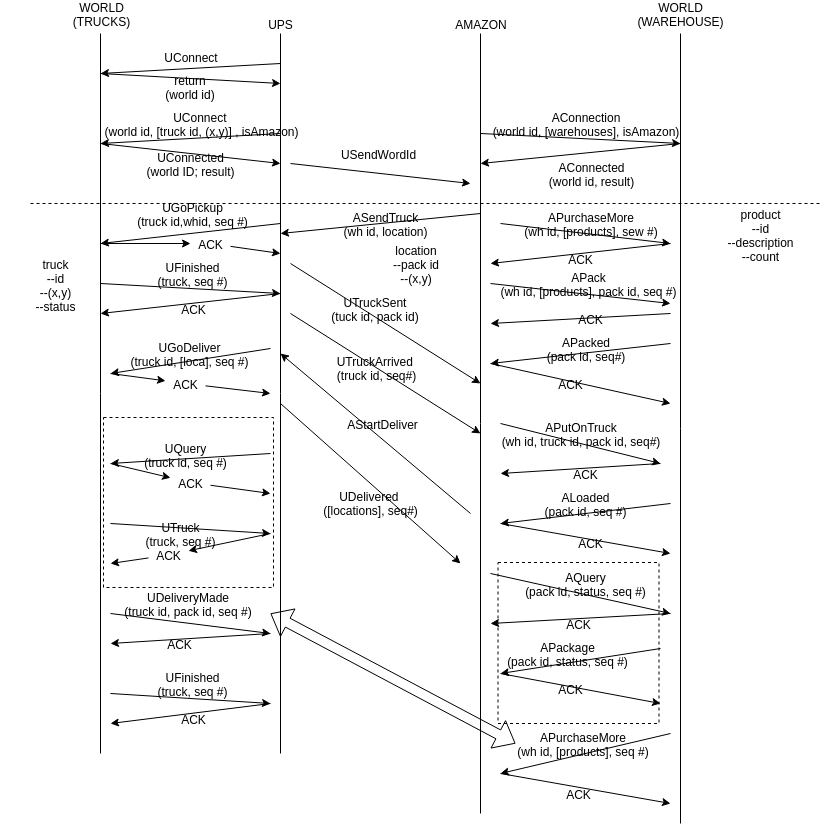
required int64 shipid=1;

required string info=2;

required int64 seqnum=3;

}

**Context Flow**



**Phase 1 -- Connected to the world**

At the starting point of the communication, the UPS server will send a UConnection command to the simulator with a blank word id, and receives the world id for the communication. Both UPS and Amazon servers are required to connect to the world with the shared world id. Also, when connecting to the world, both servers are supposed to specify their resources: UPS has trucks {id, location(x, y), status}, while Amazon has warehouses {id, location(x, y)}}

**Phase 2 -- Handling Request**

Consider a context, when a customer want to buy table from Amazon. After the order is made, Amazon begins with packing the table and calls a truck to deliver the table. Amazon needs to tell where is the warehouse (wh id) who sufficient inventory, and both track number and address of destination for UPS to keep track of the delivery.

UPS then picks an availabe truck (truck id) and tells it where to go (wh id). At the same time, UPS should inform Amazon of the truck(truck id) sent for that package (package id). When truck arrives the specified warehouse and ready to load (some packages are packed), the world will send a UFinished response to UPS with the truck id who is ready for loading.

When the truck is ready and the table is packed, Amazon issues a command to load the table onto the truck. As long as the loading is done, Amazon will delivery a message to UPS to go delivery. After receiving the signal, UPS issues UGoDeliver command the world with truck id and location info {pack id, dest (x, y)}. From this point on, customer could query the truck info with the truck id on UPS website, and query the package status with pack id on Amazon website. The world would reply Utruck {id, real-time location (x,y), status}, and APackage {pack id, status} respectively. When A package successfully reaches the destination, the world will respond UDeliveryMade {truck id, pack id} to UPS, and APurchaseMore {wh id, [products]} to Amazon. Once all of the packages are delivered, UPS would receive a UFinished from the word with the truck info {id, real-time location (x,y), status}.