CSCI 4220 Lab 5

Lab 5: gRPC

Due Wednesday October 14th

Setting up gRPC with Python

The following instructions are for Linux/WSL, you can find instructions for other platforms as well as these instructions here.

Python module install

First, make sure pip is up to date by running:
`python3 -m pip install --upgrade pip --user`

If the above command fails due to `/usr/bin/python: No module named pip` then you must first run:
`sudo apt install python3-pip`

If you get 404 errors trying to run the apt install, first run:
`sudo apt update`

The `--user` in the next few steps ensures that you can run the module without installing it system-wide. Install the base module by running:
`python3 -m pip install grpcio --user`

Next install the gRPC tools needed for handling `protoc`:
`python3 -m pip install grpcio-tools googleapis-common-protos --user`

Finally, navigate to a directory that's easy to reach, and then download the example code:
`git clone -b v1.32.0 https://github.com/grpc/grpc`

To rebuild/regenerate the gRPC code for the helloworld example from inside `grpc/examples/python/helloworld` run the following command, in the PDF it is two separate lines but you must run it as one command (be careful when copy-pasting):
`python3 -m grpc_tools.protoc -I../../protos --python_out=. --grpc_python_out=. ../../protos/helloworld.proto`

To run the helloworld program, from inside `grpc/examples/python/helloworld`, in one terminal run `python3 greeter_server.py` and then in a second terminal run `python3 greeter_client.py`

Lab continues on next page

Modifying the Route Guide Example

For this lab, you will modify the code in `examples/python/route_guide_server.py`, `examples/python/route_guide_client.py`, and `examples/protos/route_guide.proto`

First, add capability for the server to remember all routes that are passed in through *RecordRoute()* . Each route should be associated with an incremental ID number decided by the server: it will simply start with ID 0 for the first route, ID 1 for the next route, and so on. Remember that a route is a sequence of Points.

Next, extend the *RouteSummary* message so that *RecordRoute()* also reports back the ID number it has assigned to the recorded route. Have the client print this new information in addition to what it was already printing in *guide_record_route()*.

Add a new method called *RouteRetrieve()* that takes in a route ID, and returns either a list of Points corresponding to the route that was recorded earlier with the given ID, or an empty list if the ID was invalid. The client should print the route ID out, and then print the received route in order. Remember that this may be a route that was recorded by a different client than the one making the *RouteRetrieve()* call.

Finally, add a new function to the client called *guide_route_retrieve()* that has some tests for your new function. Make sure to modify *run()* to call your tests!

Submission

Submit your team's solution as two Python files called `route_guide_server.py` and `route_guide_client.py`, as well as the modified proto file `route_guide.proto`. The TAs will grade your code manually, there is no autograding beyond checking that the files exist in your submission.