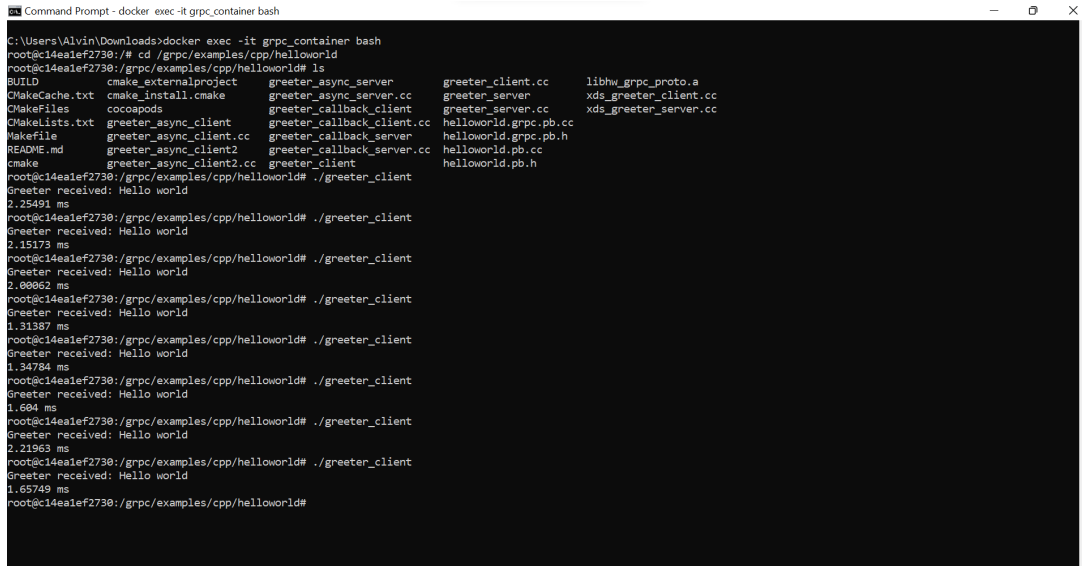


Lab 1 - RPC

1. What was the round-trip time (RTT) or latency of a message between one gRPC client and one gRPC server?

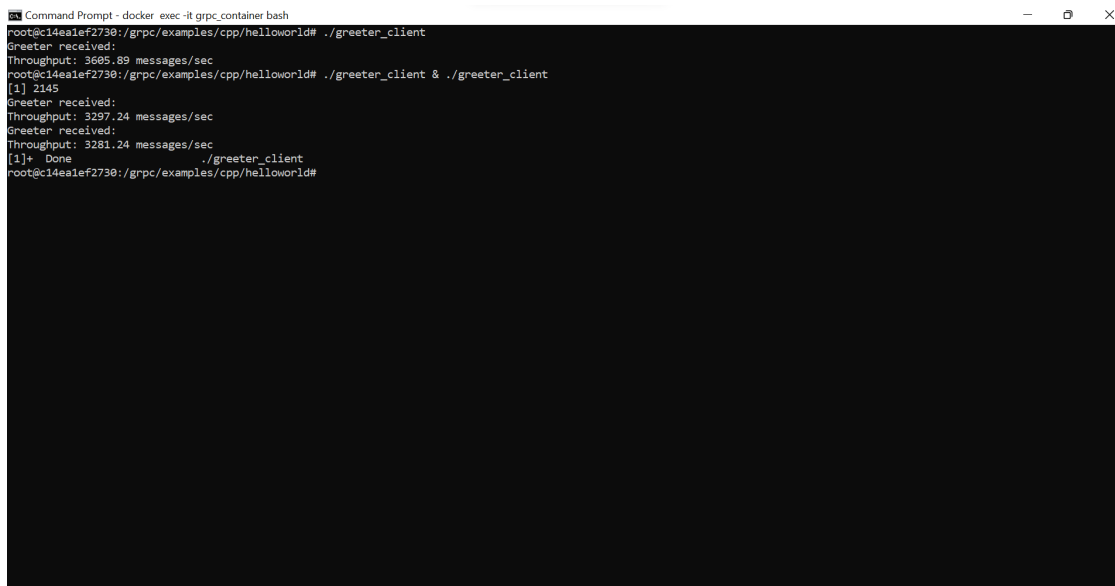


```
Command Prompt - docker exec -it grpc_container bash
C:\Users\Alvin\Downloads>docker exec -it grpc_container bash
root@c14ea1ef2730:/# cd /grpc/examples/cpp/helloworld# ls
BUILD                  cmake_externalproject greeter_async_server  greeter_client.cc    libhw_grpc_proto.a
CMakeCache.txt        cmake_install.cmake   greeter_async_server.cc greeter_server        xds_greeter_client.cc
CMakeFiles            cocopods              greeter_callback_client greeter_server.cc     xds_greeter_server.cc
CMakeLists.txt        greeter_async_client  greeter_callback_client.cc helloworld.grpc.pb.cc
Makefile              greeter_async_client.cc greeter_callback_server helloworld.pb.cc
README.md             greeter_async_client2 greeter_callback_server.cc helloworld.pb.cc
cmake                 greeter_async_client2.cc greeter_client         helloworld.pb.h

root@c14ea1ef2730:/grpc/examples/cpp/helloworld# ./greeter_client
Greeter received: Hello world
2.25491 ms
root@c14ea1ef2730:/grpc/examples/cpp/helloworld# ./greeter_client
Greeter received: Hello world
2.15173 ms
root@c14ea1ef2730:/grpc/examples/cpp/helloworld# ./greeter_client
Greeter received: Hello world
2.00062 ms
root@c14ea1ef2730:/grpc/examples/cpp/helloworld# ./greeter_client
Greeter received: Hello world
1.31387 ms
root@c14ea1ef2730:/grpc/examples/cpp/helloworld# ./greeter_client
Greeter received: Hello world
1.34784 ms
root@c14ea1ef2730:/grpc/examples/cpp/helloworld# ./greeter_client
Greeter received: Hello world
1.684 ms
root@c14ea1ef2730:/grpc/examples/cpp/helloworld# ./greeter_client
Greeter received: Hello world
2.21963 ms
root@c14ea1ef2730:/grpc/examples/cpp/helloworld# ./greeter_client
Greeter received: Hello world
1.65749 ms
root@c14ea1ef2730:/grpc/examples/cpp/helloworld#
```

As shown by my screenshot of my terminal above, it seems that the latency between one gRPC client and one gRPC server on my machine varies between 1-2 milliseconds.

2. What was the throughput (i.e., requests/sec or messages/sec) of one gRPC server when one gRPC client is running and when two gRPC clients are running?



```
Command Prompt - docker exec -it grpc_container bash
root@c14ea1ef2730:/grpc/examples/cpp/helloworld# ./greeter_client
Greeter received:
Throughput: 3605.89 messages/sec
root@c14ea1ef2730:/grpc/examples/cpp/helloworld# ./greeter_client & ./greeter_client
[1] 2145
Greeter received:
Throughput: 3297.24 messages/sec
Greeter received:
Throughput: 3281.24 messages/sec
[1]+  Done                  ./greeter_client
root@c14ea1ef2730:/grpc/examples/cpp/helloworld#
```

It seems that the throughput with one gRPC client running is about 3600 messages/sec, and the throughput with two gRPC clients running is about 3300 messages/sec.

3. **A brief but detailed explanation of how you went about measuring the latency and throughput. You should include any decisions you made (e.g. wall clock vs. CPU clock, synchronous vs. asynchronous RPCs).**

To measure the latency between one gRPC server and one gRPC client, I just used the C++ chrono library which is common for measuring execution time, and put the timepoints before and after the RPC function call within the greeter_client.cc file then printed the time duration to console. I believe this is a method of CPU clock since it directly measures the time of the execution process.

```
113 auto t1 = high_resolution_clock::now();
114 std::string reply = greeter.SayHello(user);
115 auto t2 = high_resolution_clock::now();
116
117 duration<double, std::milli> ms_double = t2 - t1;
118 std::cout << "Greeter received: " << reply << std::endl;
119 std::cout << "Latency: " << ms_double.count() << " ms" << std::endl;
```

To measure the throughput of one gRPC server and gRPC clients, I ran the RPC function call through a loop of 1000 to simulate multiple messages between the gRPC server and gRPC client. Since we have 1000 messages, we can calculate *messages per second* from dividing 1000 by the total amount of seconds it takes to finish executing the process of 1000 messages. Then we repeat this process with running two clients at the same time as shown with the terminal commands above for question #2.

```
113 auto t1 = high_resolution_clock::now();
114 for (int i = 0; i < 1000; ++i) {
115     std::string reply = greeter.SayHello(user);
116 }
117 auto t2 = high_resolution_clock::now();
118
119 duration<double, std::milli> ms_double = t2 - t1;
120 double throughput = 1000 / (ms_double.count() / 1000); // convert ms to sec
121
122 std::cout << "Greeter received: " << reply << std::endl;
123 std::cout << "Throughput: " << throughput << " messages/sec" << std::endl;
```

To finish answering this prompt, we are using a synchronous RPC based on evidence that we could see from the greeter_server.cc file.

```
// Register "service" as the instance through which we'll communicate with
// clients. In this case it corresponds to an *synchronous* service.
builder.RegisterService(&service);
```