Distributed Computing Project Part 2

Conor Hartigan | T00205418

1. Describe a redesign of your short message service as a microservice using RPC or REST as an interface – explain why you choose the interface method you chose (REST/RPC) and describe the microservice principles you will apply in your design. Using pseudo-code describe the implementation of the four methods of your service: **15 marks**

In this Project I have decided to use ONE RPC for the interface. RPC is action orientated making it more efficient for procedures and commands such as the login and logout functions whereas REST is more optimized towards getting, which would be more suitable for functions such as upload and download.

Another benefit of using ONE RPC is the ease of setup, as shown below there is minimal effort needed in the setup phase of the project, whereas REST requires a custom client which would be extremely time consuming by comparison.

1. Implement **one** of the four methods (**15 marks**)

Out of the four methods I chose to implement the login function using ONC RPC. To start with I wrote the IDL file to generate the two required files.

Text

Description automatically generated

Fig 2.1 Echo.x used to generate ECHOPROGServer and ECHOPROG

1. Compare and contrast your original protocol-based solution with the new microservices solution **20 marks (**make four points supported with examples**)**

The first thing I noticed during this project is that RPC is far easier to implement. Previous approach required datagrams and sockets which required a lot of work for basic functionality. With the RPC approach I used this was not the same situation. RPC is relatively easier to implement. RPC only requires an easy to write IDL file with parameters and methods required for the implementation of the project. Once compiled using the “java Jrpcgen -S < echo.x” command the two required files for setup were autogenerated. There was also no need to go through the great effort of designing the protocols and their codes. To sum up my point, the RPC setup was more efficient and far less time consuming than the implementation of the previous protocol-based solution.

While working on both projects I kept the idea of redesign in mind, the previous protocol-based solution would require considerably more time to refactor, this would also present more of an issue in terms of scalability as the requires for a project such as this would require more of a rework as opposed to the RPC design. For the RPC design this would not present itself as as big an issue since all the work that would be required is the rewriting of the IDL file to recreate the automatically generated classes to be implemented in the project.

While with the original protocol-based solution it can be beneficial to have to write the protocol design document so that the programmer can have a greater understanding of all the requirements that are necessary prior to starting the project it can be time consuming to write the document. In general, an agile approach is more commonplace nowadays, as such having a protocol design document that is constantly being worked would likely prove itself more of a hinderance than a benefit. Since RPC does not require the protocols to be designed prior to commencing the project it would likely be more beneficial to a modern team of developers engaging in an agile approach.