Protocol specification

TABLE OF CONTENTS

1 Background

2 Defining the ECHO RPC

To begin, you will be modifying an existing protocol specification to define a new remote procedure.

Background

To be able to send RPC requests from the client to the server, we need to specify how to serialize and deserialize both client requests and server replies.

`rpcgen` makes this easier by taking a message protocol defined using the XDR specification language and autogenerating the necessary stubs.

Let's take a look at the current protocol specification, which can be found in `kv_store.x`. As you can see, XDR syntax is very similar to C, but keep in mind that they are two different languages (hence the `.x` file extension). We will note any differences that are relevant to this lab.

```
program KVSTORE {
        version KVSTORE_V1 {
                int EXAMPLE(int) = 1;
        } = 1;
} = 0x20000001;
```

The outer `program KVSTORE` block specifies the service that we would like to create (specifically, a key/value store). This service may have multiple versions, one of which is defined using the `version KVSTORE_V1` block contained within. This versioning feature comes in handy since backward compatibility is critical in distributed systems, where different machines may be on different versions of the protocol at a given point in time.

However, the most important section of the specification is the innermost contents, which specify the API exposed by the RPC service. Specifically, this `KVSTORE` service currently only exposes an

`EXAMPLE` RPC that takes in an integer and outputs an integer.

You may notice that there are numbers after each block. The numbers after the functions are essentially identifiers for each function. In the above example, since `EXAMPLE` is assigned the number 1, the server knows to execute the `EXAMPLE` RPC whenever it is asked to execute function 1 by the client. The number following the version block allows you to specify multiple versions of this application, which may come in handy for a production system where you might want to support older versions of your software.

> You don't need to fully understand what these field numbers are doing since you will not be deploying different versions of this assignment, but just make sure you use unique numbers to distinguish between different RPCs (i.e. the second RPC you implement should be assigned the number 2, the third should be assigned 3, and so on).

Defining the ECHO RPC

Modify `kv_store.x` to include an `ECHO` RPC, where a client sends a message string to the server, and receives that same message string back in the server's reply. If you have not already, it may help to take a look at the `rpcgen` tutorial (carefully read up to, but not including, Example 3-2 for all of the information relevant to this section).

You will be implementing this RPC's functionality in the next part.
