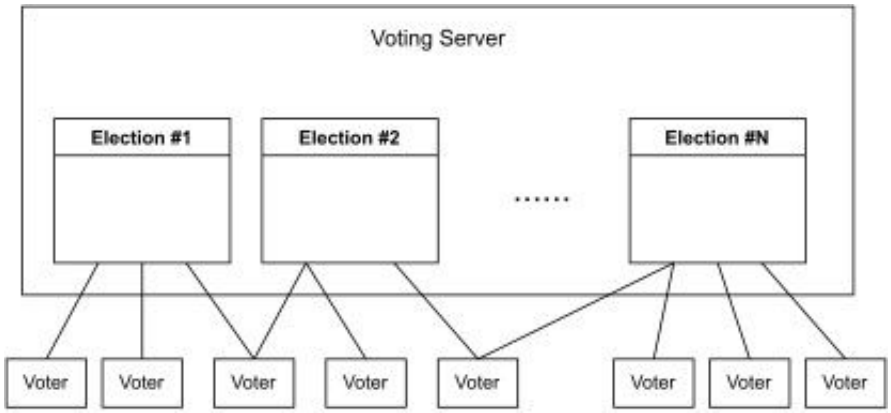
(FTC) Online Voting – Part I. RPC Interface (due 3/27)

The goal of the project is to develop an online voting system. On the system, a registered user can hold an election and collect ballots from the registered users. At the end of an election, the system will count the ballots and announce the results.

Here we define the operations that should be supported by the online voting system. The overall architecture of the system is shown in the following figure. The voting server has a list of ongoing elections. Each voter (registered user) runs a voter client to participate in an election. The volter clients cast the ballots, and the server counts the ballots at the end of each election.



In this part of the project, you will implement the communication interface between the clients and the server. To facilitate the inter-operations among the servers and the clients from each student, we will use Google gRPC for the communcatoin and Protocol Buffers to serialize/deserialize the messages. gRPC is a Remote Procedure Call (RPC) framework. Assume that program X and program Y run on two separate machines that are connected by the network. The RPC framework allows program X to invoke functions in program Y. The saves the troubles of network programming. A good starting point for learning gRPC is gRPC Basic Tutorial.

Following are the RPC APIs you need to implement for the voting server and the voting client.

```
syntax = "proto2";
package voting;
import "google/protobuf/timestamp.proto";
```

A. Local Server API

```
RegisterVoter(Voter) returns (Status)
UnregisterVoter(VoterName) returns (Status)
```

A.1. Message Formats

```
message Voter {
    required string name = 1;
    required string group = 2;
    required bytes public_key = 3;
}
```

```
message VoterName {
    required string name = 1;
}
```

```
message Status {
  required int32 code = 1;
}
```

B. RPC APIs

```
service eVoting {
    rpc PreAuth (VoterName) returns (Challenge);
    rpc Auth (AuthRequest) returns (AuthToken);
    rpc CreateElection (Election) returns (Status);
    rpc CastVote (Vote) returns (Status);
    rpc GetResult(ElectionName) returns (ElectionResult);
}
```

B.1. Message Formats

```
message Challenge {
  required bytes value = 1;
}
```

```
message Response {
  required bytes value = 1;
}
```

```
message AuthRequest {
    required VoterName name = 1;
    required Response response = 2;
}
```

```
message AuthToken {
  required bytes value = 1;
}
```

```
message Election {
    required string name = 1;
    repeated string groups = 2;
    repeated string choices = 3;
    required google.protobuf.Timestamp end_date = 4;
    required AuthToken token = 5;
}
```

```
message Vote {
  required string election_name = 1;
  required string choice_name = 2;
  required AuthToken  token = 3;
}
```

```
message ElectionName {
  required string name = 1;
}
```

```
message VoteCount {
  required string choice_name = 1;
  required int32  count  = 2;
}
```

```
message ElectionResult {
    required int32 status = 1;
    repeated VoteCount counts = 2;
}
```

At this moment, you only need to make sure the RPC interfaces (client can make calls to server) work correctly. You do not need to worry about the program logics behind the APIs at this moment.

Please prepare the following for submission

1. The source code of your server and client implementation.

2. A README describing how to build and run your code
    a. Preferablly, you can packege the code and the build environment in a docker container in case of any depedency issues that may prevent the TA from running your code for grading
3. Write a report with the following items
    a. Simple usage instruction of your client and server. In particular, you need to explain how to customize the IP address / port numbers as TA' s network environment may not be exactly identical as yours.
    b. Screenhots with simple descpriton to demonstrate that your client and the sever can communicate property via the gRPC calls defined in the above.
    c. Explain what will happen when a gRPC call ecounters network connection problems.

For questions regarding the project, please contact TA <m2955121314.11@nycu.edu.tw>