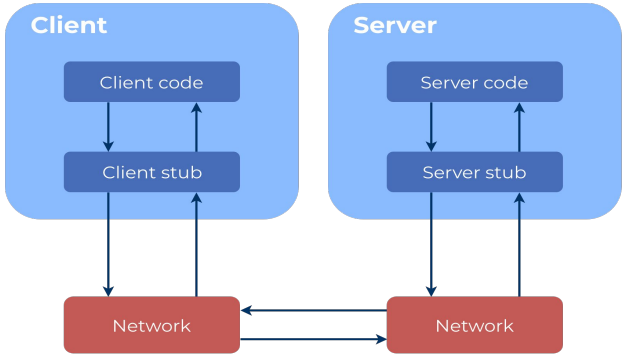

# gRPC Demo

COMS 4156

# What is an RPC (library)

- A mechanism to call a function on a different computer
- The RPC library (ideally) abstracts away all details of network communication, and provides an interface similar to a local function
- Server defines the **API** and **logic** of a function, and registers it
- Client uses a local stub to call the function on the server through that API



# What is gRPC

- Google’s open-source RPC framework
- Supports C++, Java, Python, Go, Ruby, Node, …
- Uses Protocol Buffers (proto or protobufs) for arguments/responses

# Protobuf One-pager

- Define structs (or messages) in language-agnostic format
- Serialize/deserialize using strings

**person.proto**

```
message Person {
  string name;
  int32 id;
  string email;
}
```

Protobuf compiler →

**Person.java**

```
public class Person {
  private String name;
  private int id;
  private string email;


  String getName() {..}
  void setName(String name) {..}

  String SerializeToString() {}
  void ParseFromString(String s) {}
}
```

**person.h**

```
class Person {
  std::string name;
  int id;
  std::string email;

public:
  std::string getName();
  void setName(String name);

  std::string SerializeToString();
  void ParseFromString(std::string s);
};
```

# Protobuf One-pager (page 2)

**PersonWriter.java**

```
public class PersonWriter {
  public static void main(...) {
    ...
    john = Person.newBuilder()
                 .setId(1234)
                .setName("John")
                .build();
    output = new FileOutputStream(args[0]);
    output.write(john.SerializeToString());
  }
}
```

**person_reader.cc**

```
int main(int argc, char **argv) {
  fstream in_file(argv[1]);
  std::string encoded_person;
  in_file >> encoded_person;

  Person john;
  john.ParseFromString(encoded_person);
}
```

# Demo

Demo repo at https://github.com/KidusAM/remote-add

# Advantages of RPC

- Define/call functions natively, without having to manage type conversions or serialization
- Usually faster than REST
- Support for streaming mode and multiplexing (HTTP/2)
- Good libraries support tracing, propagation, logging, ACLs (security), discovery etc. out of the box with minimal effort
- Video about Square’s migration from a REST/Monolith to RPC/microservice architecture : https://youtu.be/-2sWDr3Z0Wo