National College of Ireland

Distributed Systems
July 2022

Ruben Lopez x20146876
https://github.com/rlopezba1/Distributed-Systems-HDSDEV_JAN22-

Contents

1 Introduction

Our company, DriveCarefully, has been commissioned to create a Cork traffic management software development application. Our client is the RSA, Road Safety Authority in Ireland and they will be able to manage the traffic from the central department through our application.

As part of the task, we will develop a set of protocols/messages and build a reference implementation that simulates the operations of an intelligent automated traffic management system. The devices/services will communicate through gRPC using the Java programming language.

There will be 3 separate services that will simulate the operations of automated and intelligent traffic systems. These operations are supported by each "service/device" and will be specified in a proto file. In addition, a simple client GUI will be developed that works as a main controller that discovers and uses the devices/services.

2 Service 1: Speed control by sections

This service will control the speed of the vehicles in a section, through the detection of the vehicle registration at the beginning of the section, at the end of the section, the calculation of the time in the section covered, the calculation of the speed and therefore know if there is speeding or not.

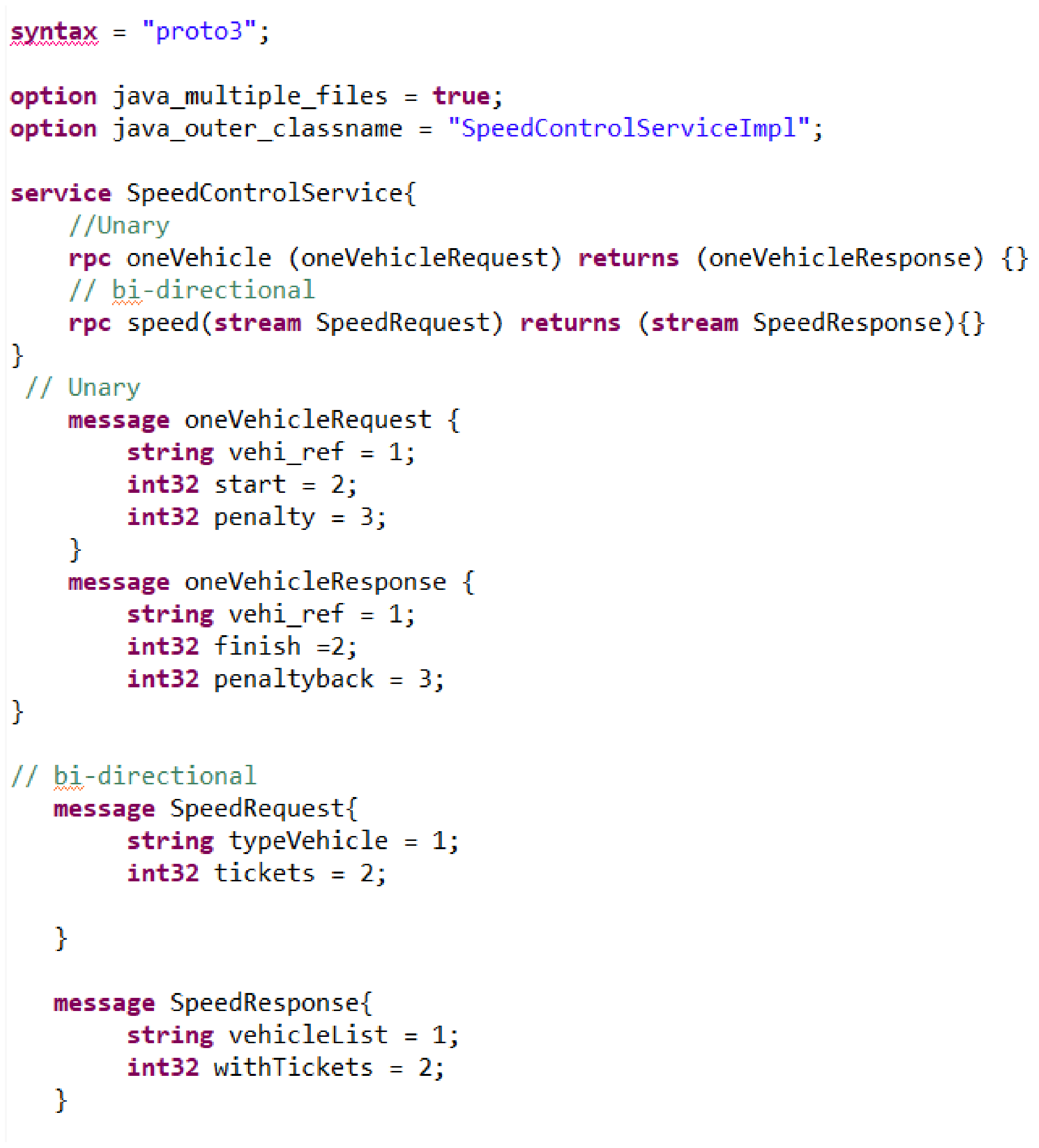
2.1 Methods

2.1.1 RPC Method 1 – Speed Control One Car

This method will be represented with **Unary**; RBA will input data about vehicle registration number and speed, and RBA will get the results back.

2.1.2 RPC Method 2 – Speed Control Data

This method will be represented **Bi-Directional Streaming**, where RSA will be able to input data of vehicles registration number, and RBA will get all the results back.



```
syntax = "proto3";

option java_multiple_files = true;
option java_outer_classname = "SpeedControlServiceImpl";

service SpeedControlService{
    //Unary
    rpc oneVehicle (oneVehicleRequest) returns (oneVehicleResponse) {}
    // bi-directional
    rpc speed(stream SpeedRequest) returns (stream SpeedResponse){}
}
 // Unary
    message oneVehicleRequest {
        string vehi_ref = 1;
        int32 start = 2;
        int32 penalty = 3;
    }
    message oneVehicleResponse {
        string vehi_ref = 1;
        int32 finish =2;
        int32 penaltyback = 3;
}

// bi-directional
   message SpeedRequest{
        string typeVehicle = 1;
        int32 tickets = 2;

   }

   message SpeedResponse{
        string vehicleList = 1;
        int32 withTickets = 2;
   }
```

3 Service 2: Traffic Signal

A system where it detects the colours of the traffic light: red, yellow, green, and when it is red, it will show "Stop", when it is orange, it will show "Start braking", when it is green, it will show "Go".

3.1 Methods

3.1.1 RPC Method 1 – Change the Traffic Lights

This method will be represented with **Client Streaming.** The RSA will be able to specify the traffic colour of the traffic lights of a specific street.

3.1.2 RPC Method 2 – Turn On/Off Street Lights

This method will be represented with **Bi-Directional streaming**, The RSA will be able to turn lights on/off a specific street.

```
syntax = "proto3";

option java_multiple_files = true;
option java_package = "grpc.examples.clientstreamtraffic";
option java_outer_classname = "TrafficSignalServiceImpl";

package traffic;

service TrafficSignalService {

  // client streaming
  rpc changeTrafficlights(stream TrafficLightsRequest) returns (TrafficLightsResponse){}

  // bi-directional
  rpc turnOnOffLights (stream LightsRequest) returns (stream LightResponse){}
  }

  // client streaming TrafficSignalService
  message TrafficLightsRequest{
    int32 TrafficLightsInput = 1;
  }

  message TrafficLightsResponse{
    string TrafficLightsOutput = 1;
  }

  // bi-directional turnOnOffLights
  message LightsRequest{
    bool turnLightsOn = 1;
  }

  message LightResponse{
    bool turnLightsOff = 1;
  }
```

4 Service 3: Crosswalk

This service will allow the RSA to see the state of the pedestrian crossings and will allow them to know the state of one.

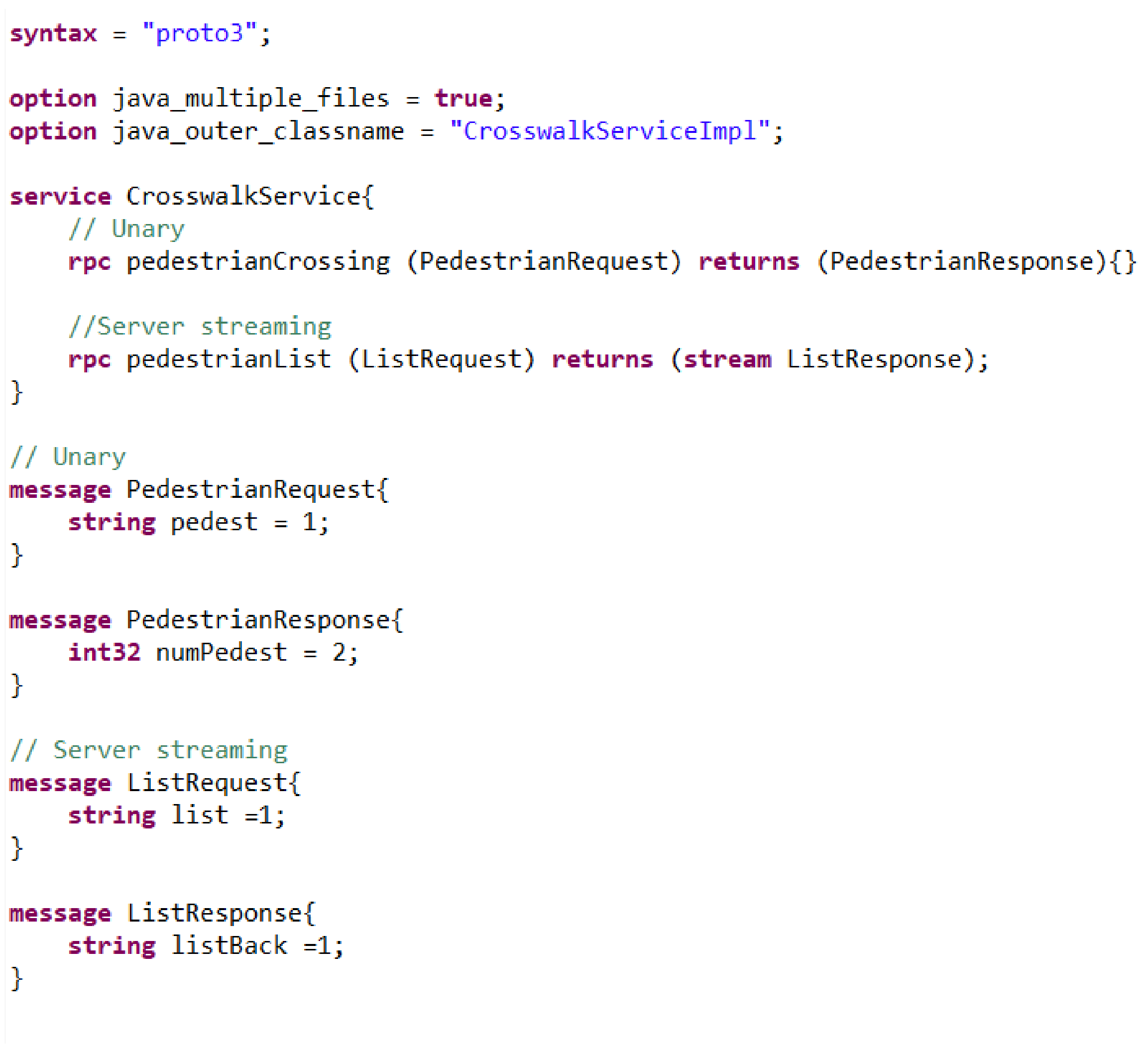
4.1 RPC Method 1

4.1.1 RPC Method 2 – Pedestrian Crossing

This method will be represented with **Unary**; RSA will be able to check status of a desired pedestrian crossing

4.1.2 RPC Method 3 – Pedestrian List

This is method will be represented with **Server Streaming**. RSA will be able to get the full list of pedestrian crossing and which of them are red or green.



```
syntax = "proto3";

option java_multiple_files = true;
option java_outer_classname = "CrosswalkServiceImpl";

service CrosswalkService{
    // Unary
    rpc pedestrianCrossing (PedestrianRequest) returns (PedestrianResponse){}

    //Server streaming
    rpc pedestrianList (ListRequest) returns (stream ListResponse);
}

// Unary
message PedestrianRequest{
    string pedest = 1;
}

message PedestrianResponse{
    int32 numPedest = 2;
}

// Server streaming
message ListRequest{
    string list =1;
}

message ListResponse{
    string listBack =1;
}
```