Distributed Systems

Project Report

Prepared for:

National College of Ireland

BSc (Honours) in Computing

Prepared by:

Wayne Hartigan

X16348136

[x16348136@student.ncirl.ie](mailto:x16348136@student.ncirl.ie)

[hartiganwayne@gmail.com](mailto:hartiganwayne@gmail.com)

4th of April 2020

# Introduction

I decided to build my Distributed Systems project around a *Smart Office*. For this project I decided to implement four services:

* Smart Lights
* Smart Thermostat
* Smart Printer
* Smart Projector

I used GRPC in Java & Python to create these services and used jmDNS to discover these services.

# Services

## Smart Lights

The smart lights service service was developed in Java and had two RPC’s attached to them.

The switchPower RPC takes in a “PowerRequest”, which has a Boolean variable names “switch”, and returns “PowerResponse” which also has a Boolean variable named switch.

This RPC was created to give the user the ability to turn on and off their smart lights.

When the server receives the request it prints out the request made and return the same variable in the response and when the client gets the response it will print out either “The lights have been turned on” or “The lights have been turned off”, depending on whether or not the response is true or false.

The changeBrightness RPC takes in a stream BrightnessRequest, which has an integer variable names brightness and returns BrightnessResponse which also has an integer variable named brightness.

This RPC was created to allow the user to dynamically change the brightness of the smart lights

The client streams a request to the server to tell the server which level of brightness t set the lights. This would be ideal for a slider bar. The server then just responds with what level of brightness the lights are currently set to which is printed on the client.

## Smart Thermostat

The smart thermostat service was developed in Java and had two RPC’s attached to them.

The switchPower RPC takes in a “PowerRequest”, which has a Boolean variable names “switch”, and returns “PowerResponse” which also has a Boolean variable named switch.

This RPC was created to give the user the ability to turn on and off their smart thermostat.

When the server receives the request it prints out the request made and return the same variable in the response and when the client gets the response it will print out either “The thermostat has been turned on” or “The thermostat has been turned off”, depending on whether or not the response is true or false.

The changeTemperature RPC takes in a TemperatureRequest, which has an integer variable names temperature, and returns a stream TemperatureResponse which also has an integer variable named temperature.

This RPC was created to allow the user to change the temperature and constantly see the response of the current temperature.

## Smart Printer

The smart printer service was developed in Java and had two RPC’s attached to them.

The switchPower RPC takes in a “PowerRequest”, which has a Boolean variable names “switch”, and returns “PowerResponse” which also has a Boolean variable named switch.

This RPC was created to give the user the ability to turn on and off their smart printer.

When the server receives the request it prints out the request made and return the same variable in the response and when the client gets the response it will print out either “The printer has been turned on” or “The printer has been turned off”, depending on whether or not the response is true or false.

The printStatement RPC takes in a stream PrintRequest, which has a string variable called statement, and returns a PrintResponse, which also has a string variable named statement.

This RPC was designed to allow the user to print a statement continuously as well as see the continuous progress of the printing.

The request sends the strings to print and the response tells the status of the printing.

## Smart Projector

The smart project service was developed in python and has one RPC attached to it.

The ChangeInput RPC takes in an Input request and returns an Input Response.

This was designed to allow the user to change the input setting on their projector. The input is sent in a request i.e. “HDMI” and the response will tell the client that the input has been changed to the selected input i.e. “The input has been changed to HDMI”.

# jmDNS

jmDNS was implemented and used to discover the services. I set up a SmartOfficeServiceDiscovery.java, SmartOfficeServiceRegistration.java as well as a GetRequest.java and HTTPServer.java.

I set up each service with a unique ID and with the discover and register files the jmDNS was able to pick up and register my services.

# gRPC

gRPC was implemented and used to create all services. Each service has a unique proto file and all four types of RPC were implemented.

# Remote Error Handling

Remote Error Handling was implemented using try catch methods on call to server to ensure errors are caught, these included RuntimeException handling.

# GUI

A GUI was not implemented, the command line must be used to invoke all services.

# GitHub

A GitHub repository was maintained from the beginning of the project with regular commits and version control.

The GitHub Repo which includes the Eclipse Java (Maven) project and the Python gRPC service can be found here:

<https://github.com/WayneHartigan/SmartOffice>