SPOT IT !

The one stop place to book a parking spot online...

Federico Cergol Pierre Francolin Karthick Mohansundaram Lasit Pant

ABSTRACT

The technological revolution has made life easy for everyone and now we are just few clicks away from booking a movie ticket, flight tickets, etc. Following this idea, we have tried with our small project to make our lives a bit more easier .
Our project revolves around a simple online parking system. One can easily check the various parking spots available pinned in the map and book them.

INTRODUCTION

The project SPOT IT ! can be used to book a parking space online. With just a few clicks away one can book a parking spot in the campus and be free from the parking hassle.
In the project we have one central server which will communicate with middle servers to see if a spot is available or not. If the spot is available then one can easily book it.

ARCHITECTURE

There are various subsystems in our architecture:

1. Every parking spot will have a small computing unit connected to it, that we simply called unit. This small box has some sensors which can poll the state of the parking spot and has some kind of actuator (LEDs/barriers) that let it signal the current state of the parking spot (empty, full, booked, unavailable...). As we couldn't really get this system for cost and deployment reason, we simply simulate units with a GUI.

2. Every parking lot will have a server which will aggregate and manage all the data from the units, called middle server. This server will run a rmi service to allow communication with its units. Moreover, this server will run a RESTful service to provide access to the relevant information from outside.

3. There will be one central server which will be the entry point for any client that wants to use this system. Middleservers will register to it. Then, when a client connects to the central server, he or she receives a list of parking lots and relevant information (ie the url for the REST interface of the middle servers). All the communications between central server and middle servers and central server and clients uses a RESTful service.

4. The client is a simple web-application that connects to central server, gets the list of parking lots available from which the user choose the best one. Finally, the client is able to contact the appropriate middle server to book a free parking spot. If there is no available parking spot, the user may be able to find another parking lots as the list contains every available parking lots within a distance of 50 km.

IMPROVEMENTS

As the sky has no limits similarly there are no limits in improving and adding functionalities to our project. In this section we have listed a few improvements that can be done:

1. Add a functionality that allows a client to book a parking spot for a specific time period and for a specific duration.

2. If the previous functionality is added, it could be nice to allow users to type in an address so that they can search for a spot before leaving. For now, we only use geolocation so that users can find a parking spot in their area. By offering both solution, we extend the potential uses of our system.
3. Once a person has booked a spot, middle servers could send back the spot details with a 4 digit code for verification.
4. A functionality could ensure that the person who parks on a reserved spot is the right one by sending to the person who reserved the spot a notification. If the spot has be stolen, the user is aware and can contact one of our server to handle the dispute on the spot.
5. Allow payment option to extend our offer to paying parkings. In a way, the system could work like the parking meter of the future.

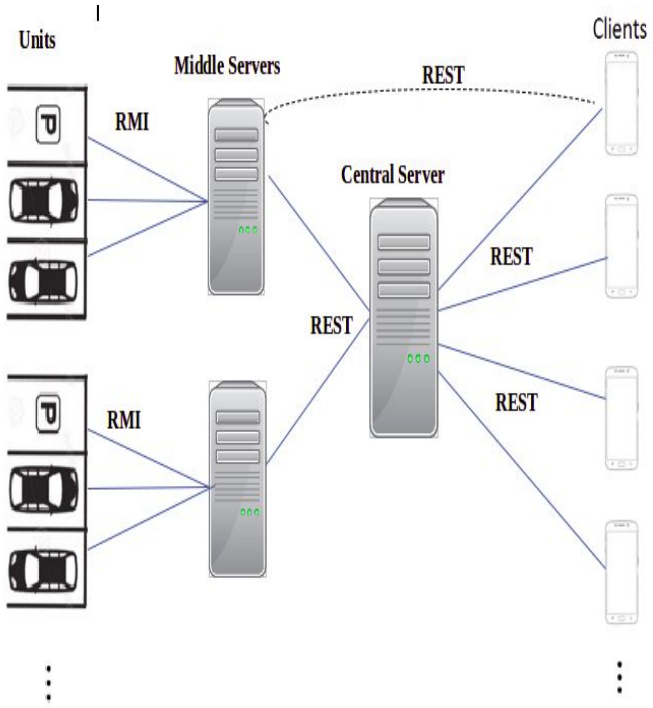
ABOUT AVAILABILITY AND SCALABILITY

Central Server

The central server, as its name suggests, plays a key role in the system by allowing communication between middle servers and the clients. If a middle server breaks down or isn't available, only this one parking lots is affected but the overall system is able to work. It's not the case with the central server as it plays such an important role in putting into relation clients and middle servers.

Robustness in the units is ensured by the rmi interface: as units become unavailable the degradation is graceful. A system for ensuring robustness in the middle server is also implemented: they continuously register with the central server and the central server will send to the clients only middle servers which have registered in say 5 minutes. Using this approach should be easy to implement a service which will check for middle servers that stops registering and will notify someone about the problem.

The availability of the all systems depends on the availability of the central server. One of the solution to guarantee availability would be to use replication of the parking lots list in different replicas of the central server. This solution would guarantee the availability as it's probable that one central server breaks down or encounter a bug but nearly impossible for for different central servers geographically scattered



around the world. This solution would also improve the scalability of the system because it would be able to spread the workload into different servers when the number of users grow rapidly and one server isn't enough to answer to all of their request. Finally, we could improve the efficiency of the system by using local replicas of central server to deal with

users. A local replica would mean that the server is closer to the client which in most cases guarantees a higher throughput of data.

Nonetheless, the replication solution would have been too complex to implement in the time we have for this project, that's why we didn't add it to the current system making the unique central server a bottleneck and a crucial point for the system.