Sure-Park Project

Architectural Design Document

Version 1.0

|  |  |
| --- | --- |
| **Team** | **Infinite Challenge(Team 3)** |
| **Members** | Namjin Lee, Jack Oh, Charles Park, Joan Kim, Jaeheon Kim |

**History**

|  |  |  |
| --- | --- | --- |
| **Ver.** | **Date** | **History** |
| 1.0 | 19, June, 2016 | Initial |
|  |  |  |
|  |  |  |
|  |  |  |

Contents

[**1.** **System Context** 3](#_Toc454113893)

[**2.** **1st Decomposition** 4](#_Toc454113894)

[**3.** **2nd Decomposition** 6](#_Toc454113895)

[**3.1) Facility Controller to Controller Service** 6](#_Toc454113896)

[**3.1.1) How to check if the Facility Controller is alive** 6](#_Toc454113897)

[**3.1.2) How to check the slot status** 7](#_Toc454113898)

[**4.** **Detail Design** 9](#_Toc454113899)

[**4.1) Facility Controller to Controller Service Packet** 9](#_Toc454113900)

[**4.1.1) Packet Structure** 9](#_Toc454113901)

[**4.1.2) Detailed Packet Scenario** 10](#_Toc454113902)

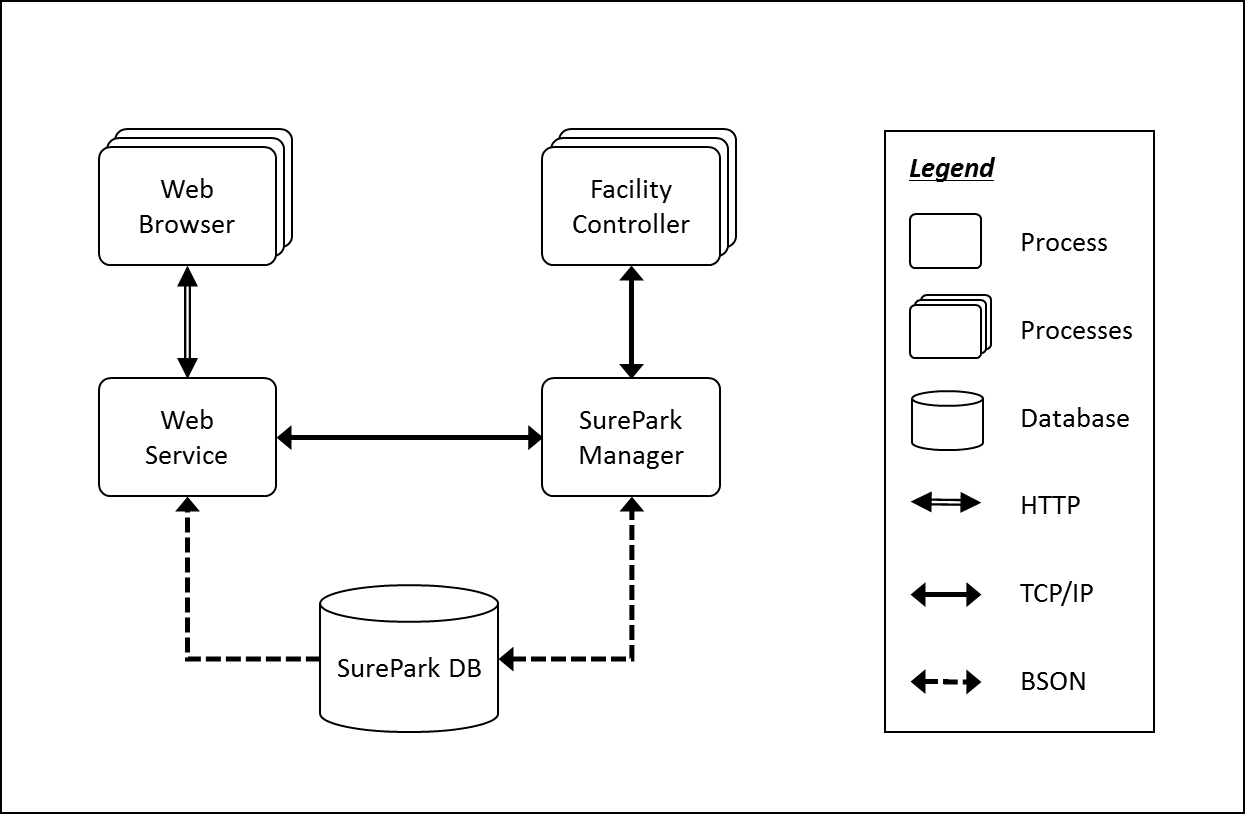
[**4.2) Controller Service Package class diagram** 11](#_Toc454113903)

# **System Context**

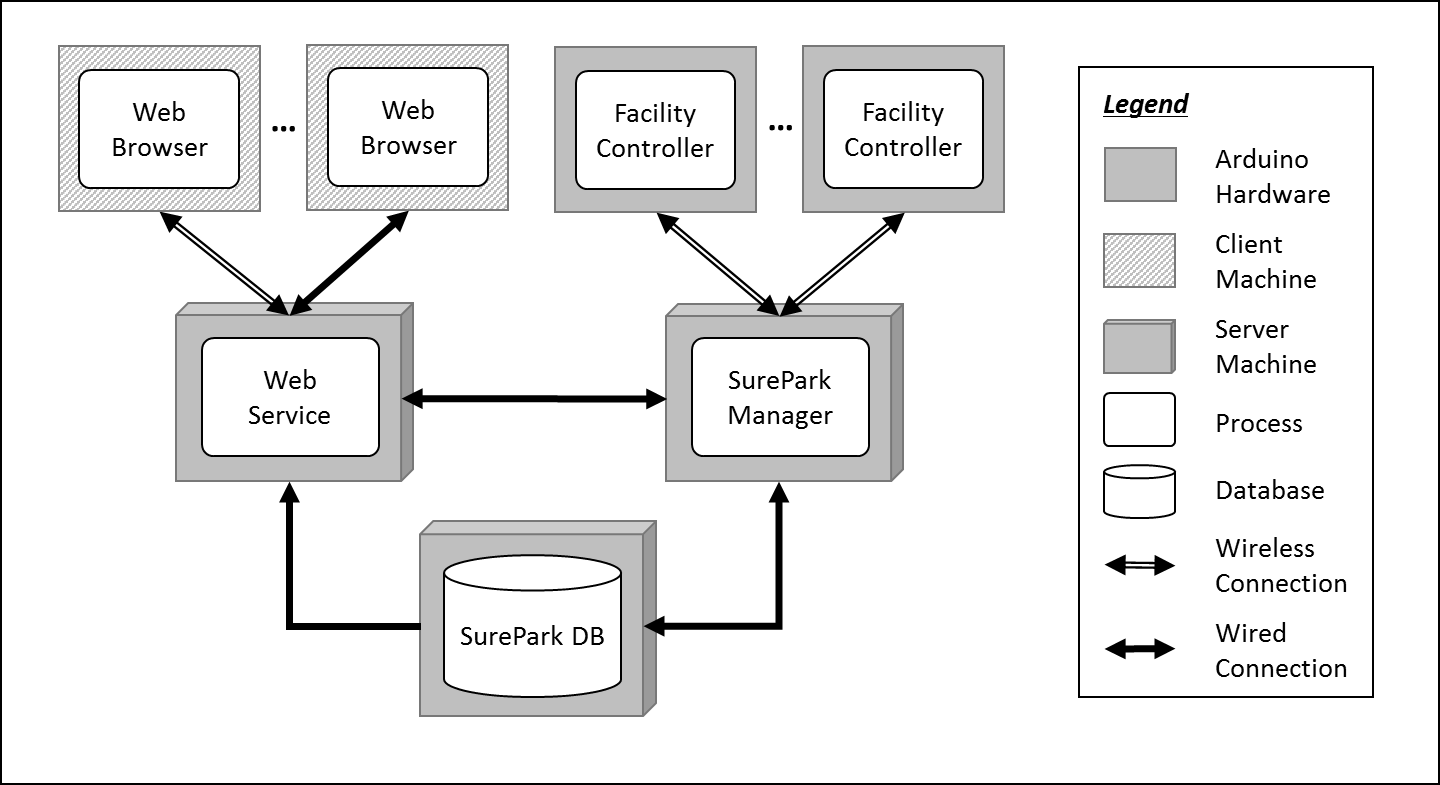
****

< Figure1. System Context >

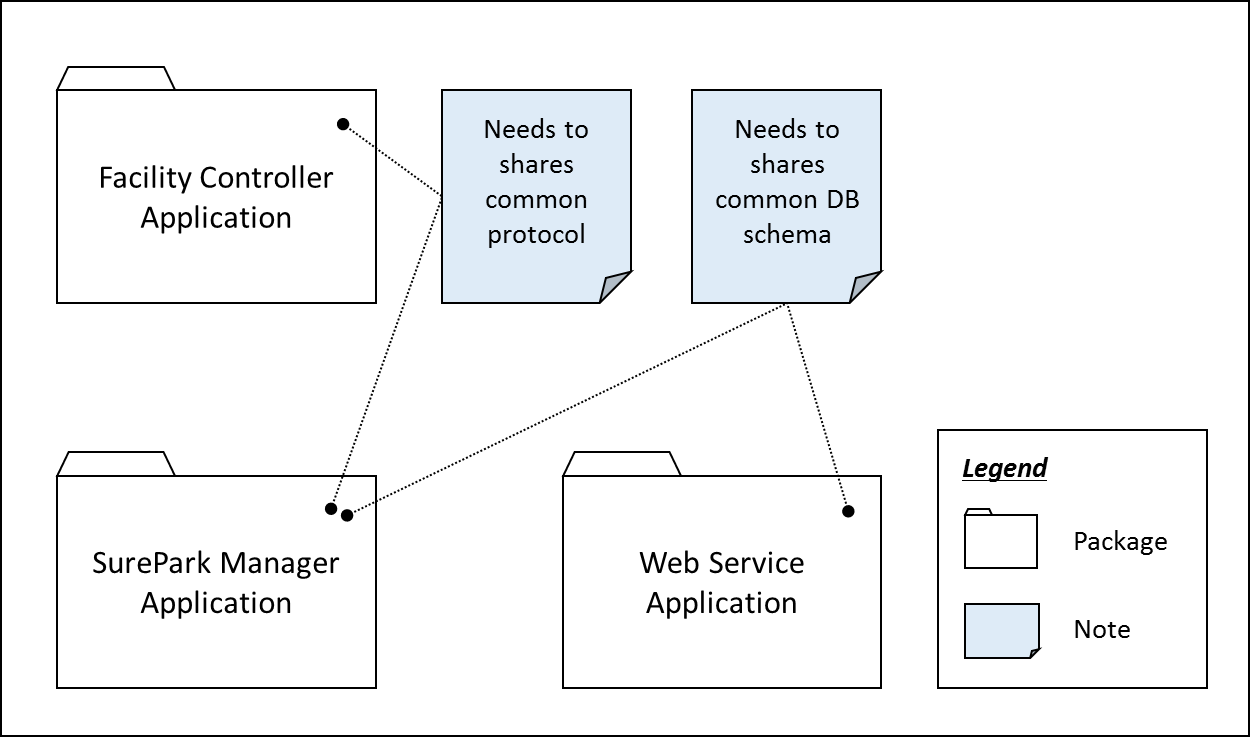
# **1st Decomposition**



< Figure2. Dynamic view of 1st decomposition >



< Figure3. Allocation view of 1st decomposition >

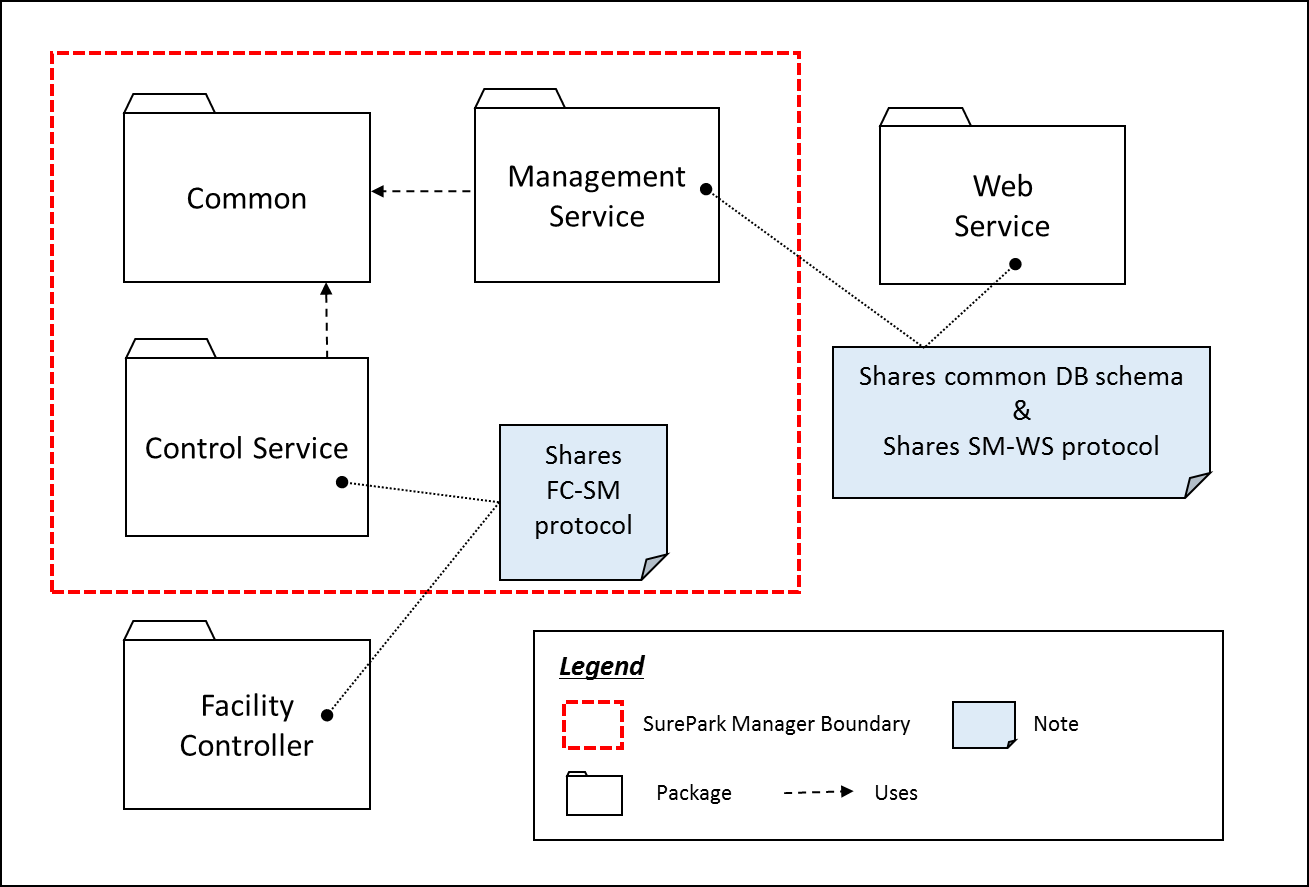


< Figure4. Static view of 1st decomposition >

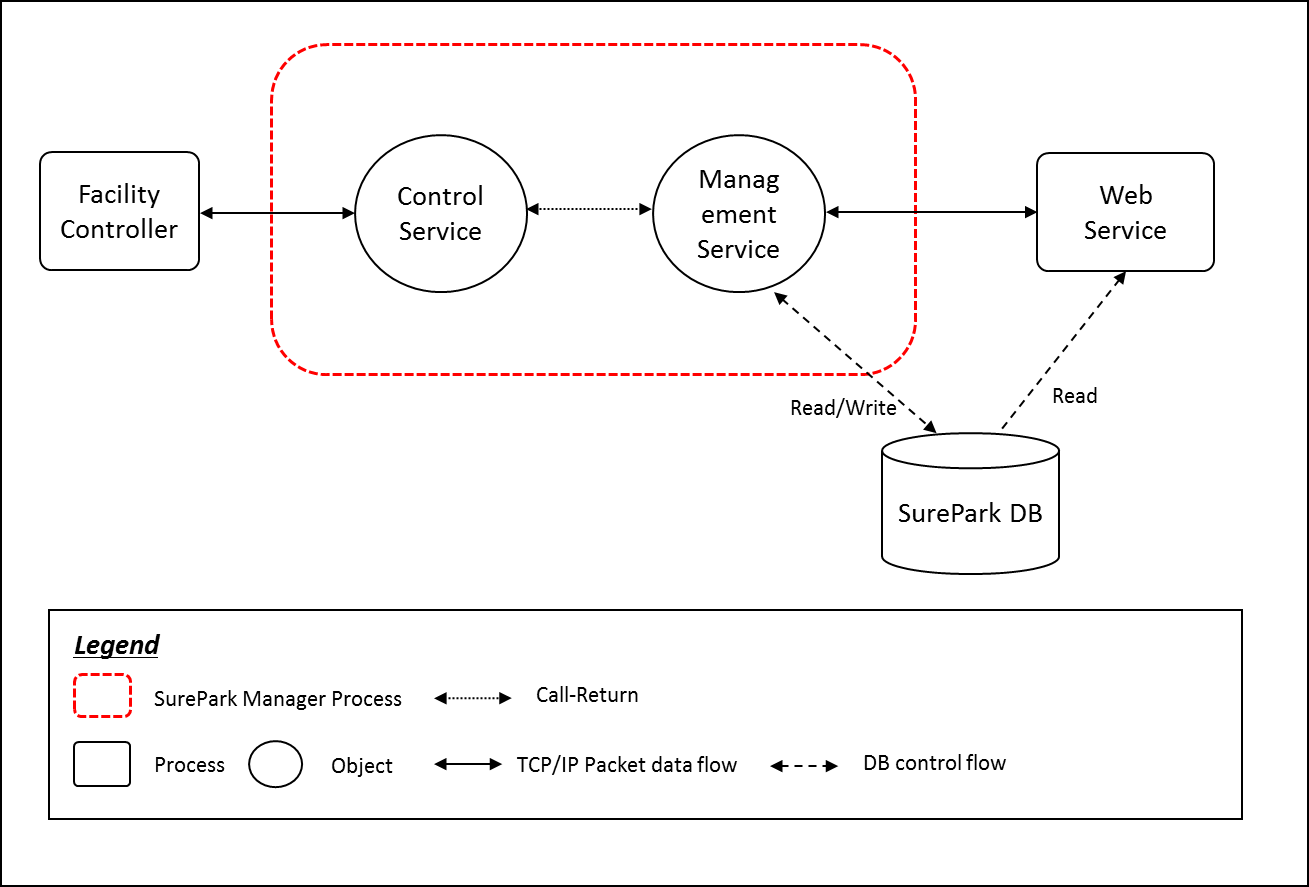
|  |  |
| --- | --- |
| **Modifiability(QA08)** | **Perspective: Dynamic** |
| Architectural Pattern | Client-Server pattern with Facility Controller and SurePark Manager.  Repository pattern with SurePark DB |
| Rationale | Modifiability is one of the most important QAs of the SurePark system. An engineer needs to scale up the system within a week. We have divided the whole system into 5 parts according to their responsibilities, and applied client-server and repository pattern to connect each parts.    < Figure6. Architectural patterns of SurePark System > |

|  |  |
| --- | --- |
| **Entity** | **Description** |
| Web Browser | Users, attendants and owner can access their own UI through the web browser provided by the web server. |
| Web Service | Provides users with the functions of sign-up, log in, reservation, monitoring facilities and/or showing parking statistics based on data retrieved from SurePark DB.  Sends information to SurePark Manager for DB updates. |
| Facility Controller | Controls parking facilities; get the status of parking slots, turn on/off LEDs, detect a car at the gates and open/close the gates.  Receives data from SurePark Manager to control LEDs and/or gates.  Sends data to SurePark Manager to update the status of parking slots. |
| SurePark Manager | Handles show-up and no-show scenarios based on DB information.  Updates SurePark DB when a user has signed up, a reservation has been made or facility status has been changed. |
| SurePark DB | Keeps all of the data about users, garages and reservations.  Only can be updated by SurePark Manager. |

# **SurePark Manager**



< Figure6. Static view of SurePark Manager >



< Figure6. Dynamic view of SurePark Manager >

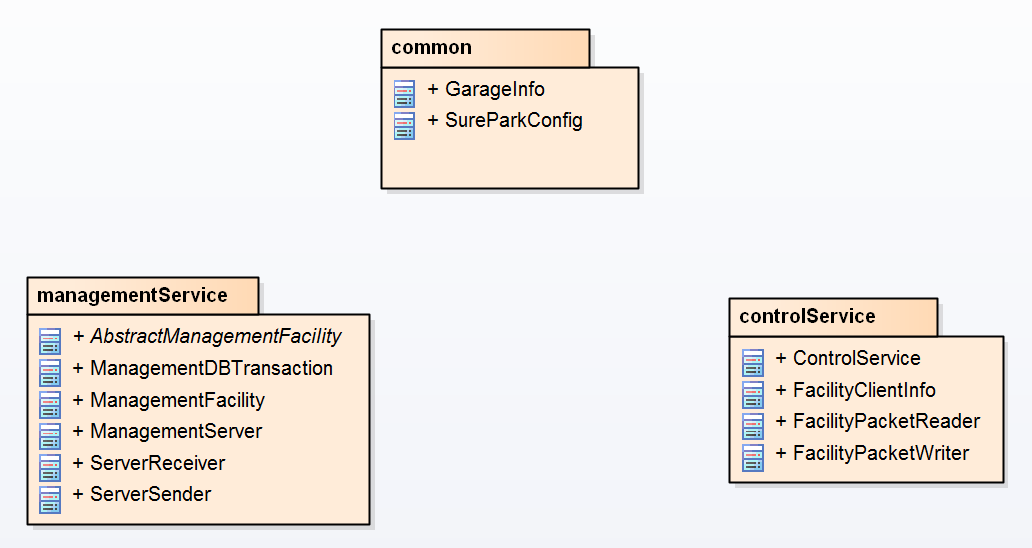
### **3.1.1) How to check if the Facility Controller is alive**

|  |  |
| --- | --- |
| **Availability(QA02)** | **Perspective: Dynamic** |
| Architectural Pattern | Client-Server structure with heartbeat tactic. |
| Rationale | Facility Controller send a packet every 10 seconds to SurePark Manager. If Controller Service doesn’t get this packet within 20 seconds, the SurePark Manager notify it to SurePark WebService for alarming to attendant for fixing the system. |

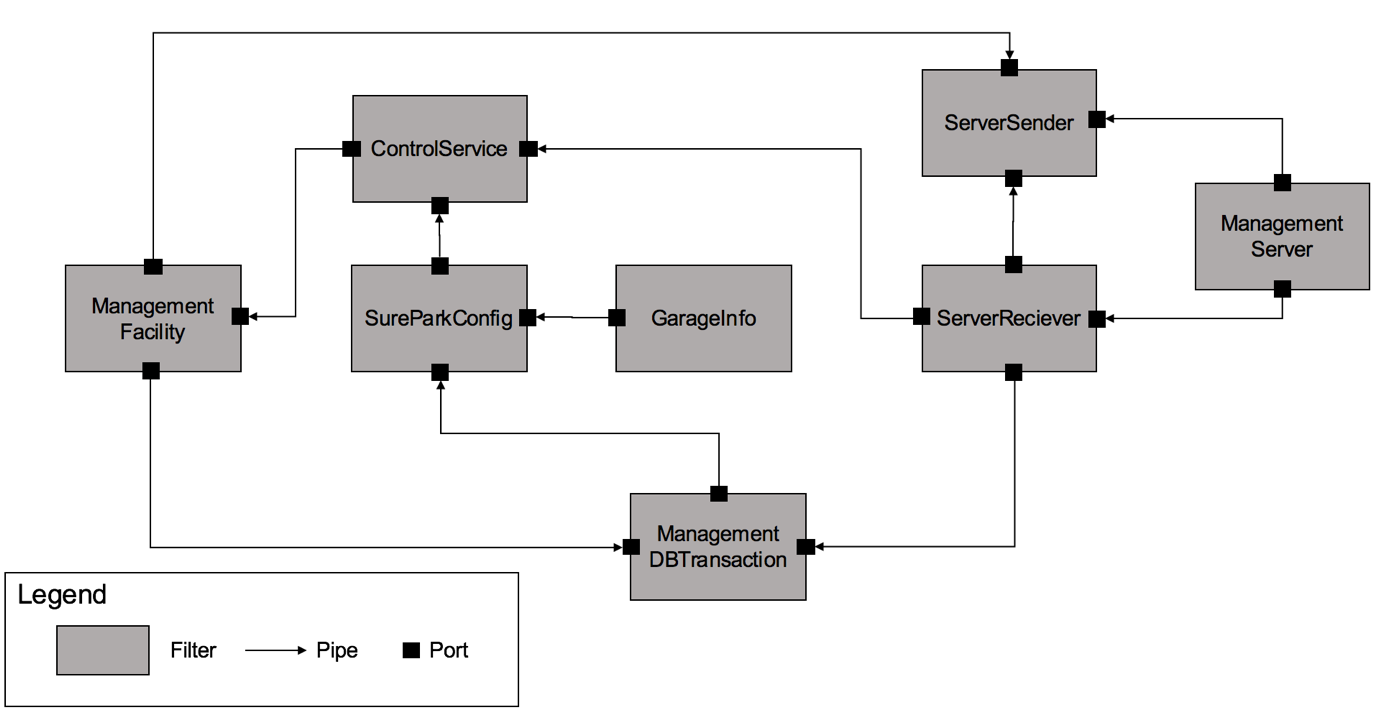


3.2) Control Service to Management Service

3.2.1) Static perspective (Package Diagram)



3.2.2) Dynamic perspective (Pipe and Filter)



3.3) Web Service

3.2.1) Static perspective (Package Diagram)



# **Detail Design**

## **4.1) Facility Controller to Controller Service Packet**

### **4.1.1) Packet Structure**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Start Symbol** | **Arduino Id** | **Code** | **Value** | **End Symbol** |
| 1byte($) | 4byte | 1byte | Variable length | 1byte(\n) |

Start Symbol: Start point of valid packet.

Arduino Id: Assigned the Arduino Id.

Code: Indicate what kind of packet is. I means “Information”. S means “Slot Status”. G means “Entry Gate”. L means “LED”.

Value: It depends on “Code”. Please refer to “detailed packet scenario”.

End Symbol: End point of valid packet.

### **4.1.2) Detailed Packet Scenario**



|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Send Information | After connection, the Controller Service has to send information to the Facility Controller.   |  |  |  |  |  | | --- | --- | --- | --- | --- | | $ | Arduino Id | I | Slot No. | \n |   Ex) $0001I4\n (Garage 1 consists of 4 stalls.) |
| Send Slot Status | Basically, the Facility Controller has to send the slot status to a Controller Service every 5 seconds. And if slot status is changed, the Facility Controller send it again regardless of under 5 minutes.  The value 0 means a slot is opened, 1 means a slot is occupied. 2 means a slot is broken.   |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | | $ | Arduino Id | S | Slot 0 | Slot 1 | ... | Slot N | \n |   Ex1) $0001S1001\n (Slot 0 and slot 3 are occupied.)  Ex2) $0001S0000\n (All slots are opened.) |
| Open Entry Gate | |  |  |  |  |  | | --- | --- | --- | --- | --- | | $ | Arduino Id | G | 1 | \n |   Ex1) $0001G1\n (request to open the entry gate.) |
| Turn on Slot LED | |  |  |  |  |  | | --- | --- | --- | --- | --- | | $ | Arduion Id | L | Slot No. | \n |   Ex1) $0001L1\n (Slot 1's LED has to get "Green".)  Ex2) $0001L0\n (Slot 0's LED has to get "Green".) |

## **4.2) Controller Service Package class diagram**



|  |  |
| --- | --- |
| ControllerService | ControllerService communicate with other package class. It creates socket and it consists of FacilityPacketReader and FacilityPacketWriter. |
| FacilityPacketReader | It’s a thread. It manages received packets from the Facility Controller. |
| FacilityPacketWriter | It’s a thread. It manages sending packets to the Facility Controller. |
| ArduinoInfo | ArduinoInfo class includes information of Arduinos. |
| Config | It’s singleton class. Other classes can access it for retrieving the Arduino information. |

4.4) Control Service to Management Service Detail design

