Project Report

BSc. Honours in Computing (4th Year)

Software Development Stream

Smart Meeting Space

Project also available on Github:

https://github.com/Virksaabnavjot/Smart-Meeting-Space

Group: I

Team Members		
Navjot Singh Virk	x13112406	Virksaabnavjot@gmail.com
Soffyan Ali	X13114531	Soffyanali@gmail.com

I. Background

NSV Smart Meeting Space is an distributed application/system. The project is an reference implementation of devices in a meeting space. For the purpose of this project naming – Laptop, Light, Mobile Phone, Printer, Projector. Where we have a service/server class for each of these devices which publishes themselves when run and the client can discover them and communicate. We have used JSON for data transfer which was done easily with the use of GSON.

II. Objectives

1. Project Problem

Describing the project problem.

We need a smart system for a building with different meeting rooms where the above listed 5 devices must be available for use by the people using the meeting room/building. Which will require implementing a distributed system/environment which will allow service discovery and communication between service and client. The devices will publish themselves and allow the following operations -

Operations Supported by each device in a smart meeting space (meeting rooms) –

Laptop – The device will need to publish its name so it can be recognised, location (which room), battery status, brightness, volume, if its switched on/sleep, charger plugged (could be more but keeping it short).

```
public Laptop(int volume, boolean switchedOn){
    this.deviceName = "Nav's Mackbook Pro";
    this.deviceLocation = "Meeting Room: SCR 3";
    Random random = new Random();
    //generating random number 1-100
    int randomNumber = random.nextInt((100 - 0) + 1) + 0;
    this.batteryStatus = randomNumber;
    this.brightness = randomNumber;
    this.volume = volume;
    this.swichedOn = switchedOn;
    this.chargerPlugged = true;
}
```

Sample constructor of Laptop device.

These are operations a laptop must be supporting and for the purpose of this prototype implementation we will actually have controls for few of the operations and few will be hard coded and same applies for other devices.

Light – The device will publish its name, location(ex. Meeting room 3, power consumption (ex. 40W), current mode (ex. Normal), brightness, switch, and different modes it supports (ex. Normal, Dark, Sunny).

Mobile Phone – The device must allow calling (and messaging), network, battery status, brightness, volume, if screen locked, plugged in, or on mute.

Printer – The device must allow printing, document selection (file name), publish it name and location, ink levels, no of copies to be printed, paper status, and printing status (ex. On hold).

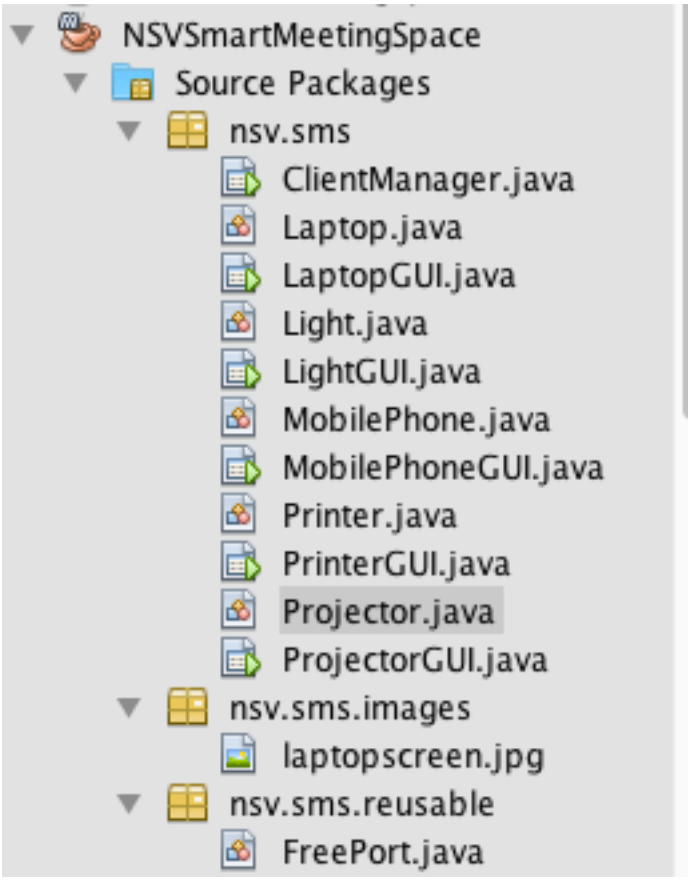
Projector – The device must allow projecting documents, resolution change, current connection type and available connection types (ex. VGA, HDMI)

2. Project Solution

We will develop a distributed system (Smart Meeting Space) using jmDNS in which the services will publish themselves for discovery and on connection they will swap service manifest (which describes how to access these services and operations they support).

III. Implementation

Project Structure



Main package: ***nsv.sms;***

ClientManager - This class manages all the client (GUI based)

//Service Classes

LaptopGUI.java

LightGUI.java

MobilePhoneGUI.java

PrinterGUI.java

ProjectorGUI.java

These are the service files for all 5 smart meeting space devices which make use of jmdns, gson, and light weight GUI

//Device Classes

Laptop.java

Light.java

MobilePhone.java

Printer.java

Projector.java

These classes describe the 5 available devices and the functionalities they support.

Package: ***nsv.sms.reusable;***

Contains file with static method which helps find free ports on the server for the use of jmdns service.

Package: ***nsv.sms.images;***

Contains image files

Important code :

//creating a JmDNS instance

```
jmdns = JmDNS.create(InetAddress.getLocalHost());
```

```
 info = ServiceInfo.create(SERVICE_TYPE, SERVICE_NAME, SERVICE_PORT, "");
```

or

Example- (creating a laptop service)

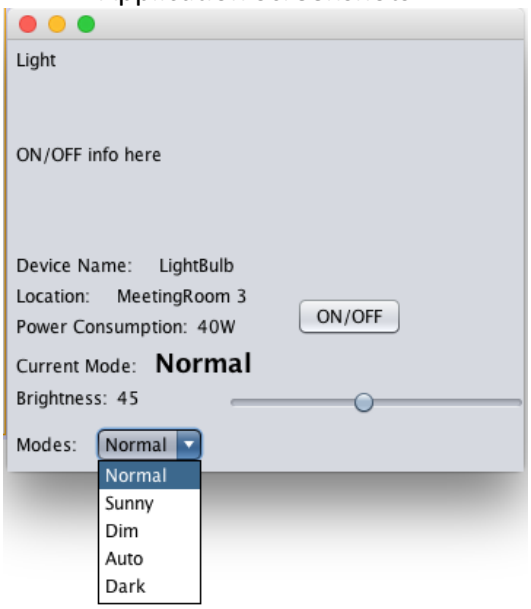
```
info = ServiceInfo.create("_laptop._udp.local.", “LaptopService, “8080”);
```

//registering the service

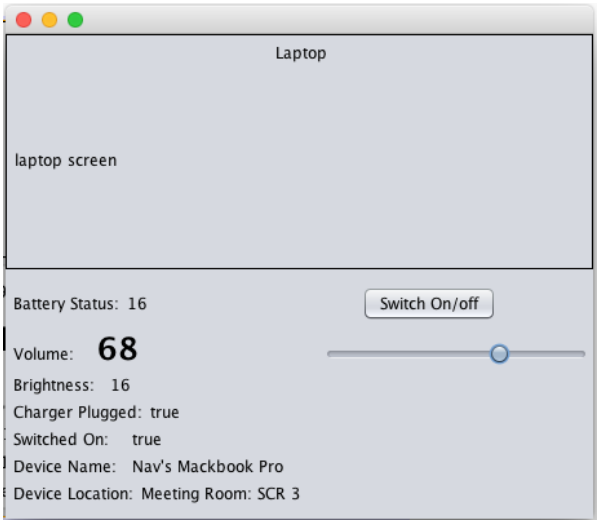
```
 jmdns.registerService(info);
```

Other - Explanation of methods, functions, class descriptions are added alongside the code.

- Application Screenshots



Light service (with a light-weight GUI and minimal control) instead of just text based UI. All other controls on client side.



Laptop service (with light-weight GUI)

References

The project is done with the help of project sample and resources available on moodle (https://moodle.ncirl.ie).

Official jmDNS was helpful while working on the project https://github.com/jmdns/jmdns

Other resources used and utilised are referenced in line with the code where used.