Distributed Systems

(Assignment-I)

Weather Monitoring System

“Storm | StormServer | StormPI”

Reg No	IT15027948
Name	R.G.D.Nayomal

Table of Contents

Table of Contents **2**

Introduction 3

Features 3

Non Functional Requirements 3

High Level Architectural Diagram (Physic al Diagram[2]). 5

Sequence diagram[1] - StormPI Weather Sensor 6

Sequence diagram - StormPI Sensor: Connecting to Storm Server 7

Sequence diagram - Sensor Registration (Associated with "Connecting to Storm PI diagram" given above) 8

Sequence diagram - RMI StormAPP and StormServer Communication 9

Requirements 10

Assumptions 10

Known Issues. 10

References 10

Introduction

“Storm” is a fully distributed weather monitoring system. It consists of mainly 3 main applications

1. Strom Server - Server which provides services to monitoring stations and sensors
    a. Socket interface - Provides communication with the sensor devices.
    b. Java RMI interface - Exposes methods which can be used by Monitoring Stations.
2. StormPI - A sensor emulator which contains
    a. Temperature sensor
    b. Pressure Sensor(Barometer)
    c. Rainfall Sensor(Rain Gauge)
    d. Humidity Sensor
3. StormApp - Client for weather monitoring,. Live weather data can be visualized in given time.

Features

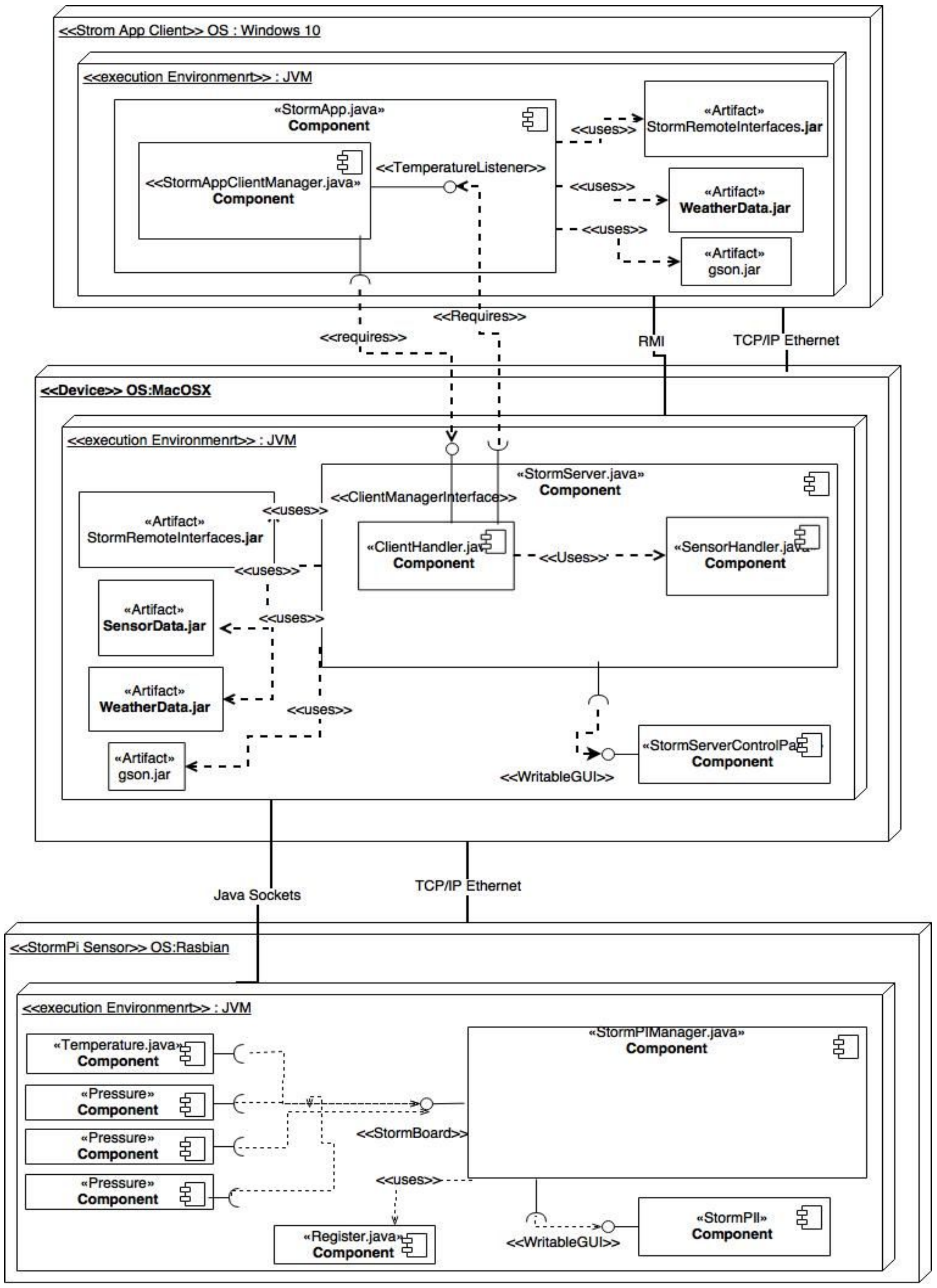
- StormApp Monitoring Stations receives live weather data and alerts on separated views.
- StormApp Supports on demand weather.
    - Monitoring Stations receives an available list of sensor locations when connected to the server.
    - User can select a location and subscribe to the weather.

Non Functional Requirements

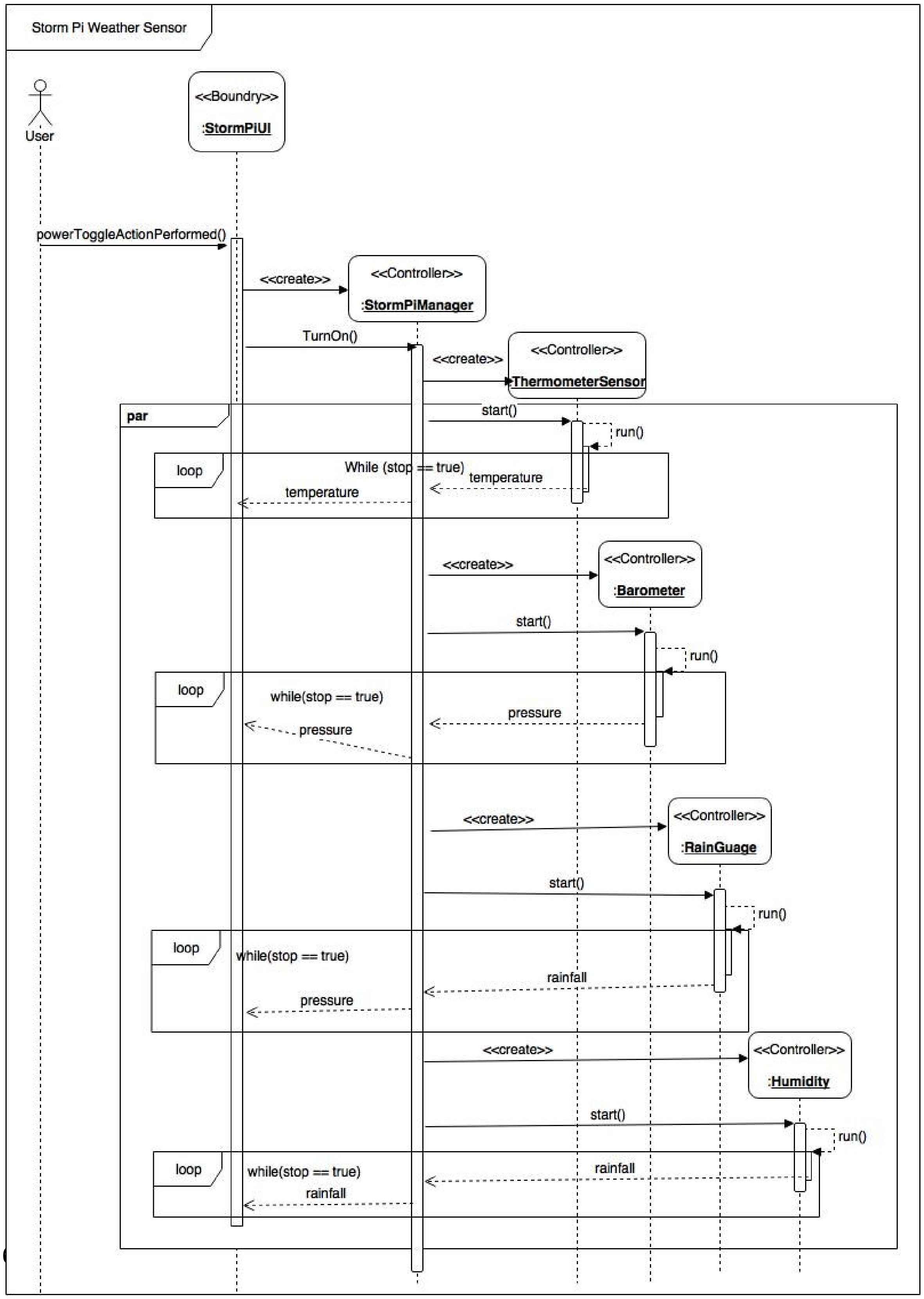
1. Storm uses the Distributed Computing principles in the architecture and the design.
    a. Architecture - Client Server
    b. Server Communicates with the Monitoring stations through the RMI interface
    c. Sensors sends weather data to Storm server through sockets.
2. Adoption of appropriate security/authentication mechanisms
    a. Sensors has to be registered with the server.
    b. Registered Data is stored in a file which is present inside the server.
    c. Monitoring Stations can log into the server only using a secured key.
3. Thread Safety
    a. Use of ConcurrentMaps to store data

   b. Use of CopyOnWriteArrayLists
   c. Object null checks before accessing the object.
   d. Synchronized file access methods
4. Performance
   a. Use of multithreaded programming in order to utilize the maximum performance in sensors client and the server

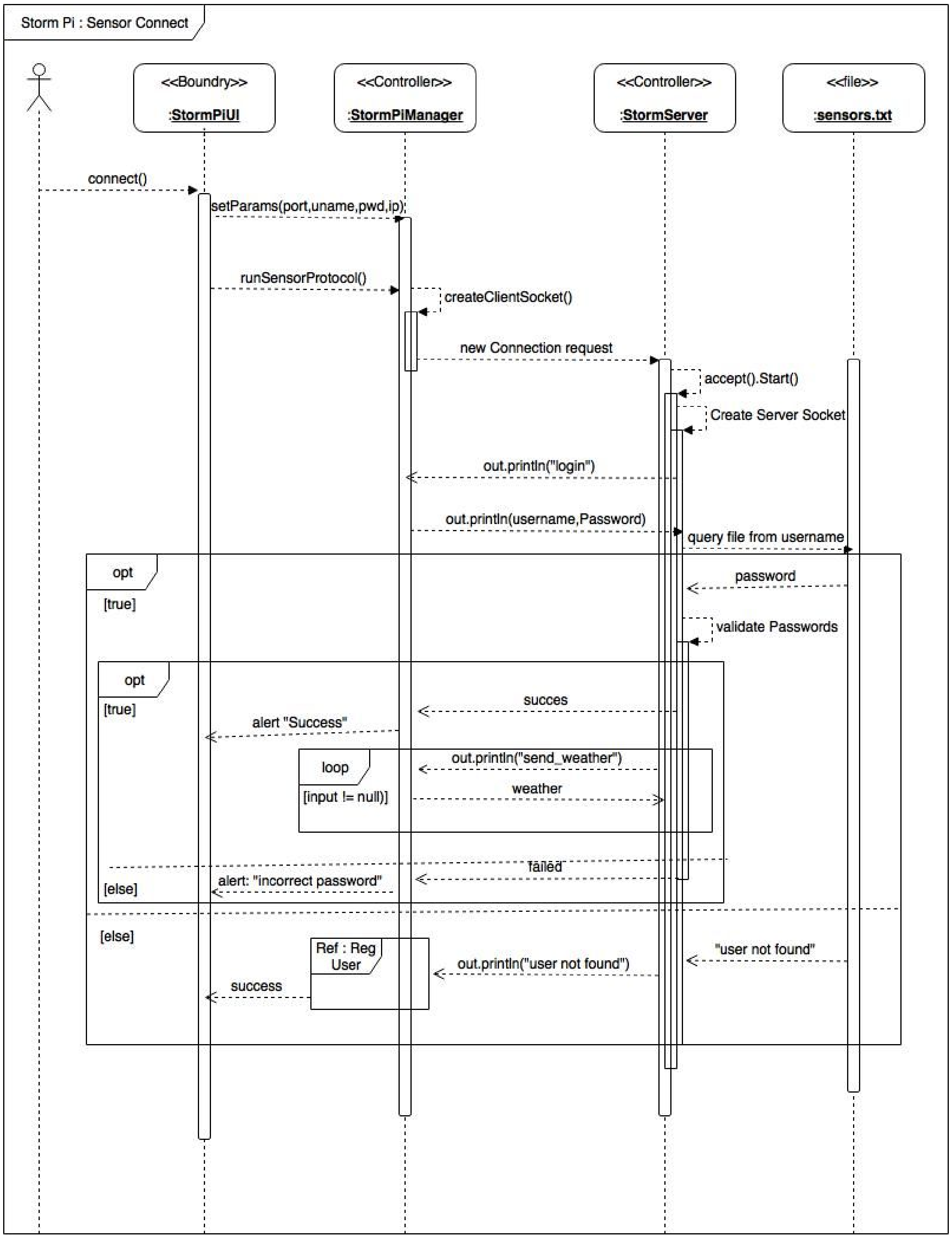
High Level Architectural Diagram (Physical Diagram[2]).


<<Strom App Client>> OS : Windows 10
<<execution Environmenrt>> : JVM
«StormApp.java» Component
<<StormAppClientManager.java» Component
<<TemperatureListener>>
<<uses>>
«Artifact» StormRemoteInterfaces.jar
«Artifact» WeatherData.jar
«Artifact» gson.jar
<<Requires>>
<<requires>>
RMI
TCP/IP Ethernet
<<Device>> OS:MacOSX
<<execution Environmenrt>> : JVM
«StormServer.java» Component
<<ClientManagerInterface>>
«ClientHandler.java» Component
<<Uses>>
«SensorHandler.java» Component
«Artifact» StormRemoteInterfaces.jar
«Artifact» SensorData.jar
«Artifact» WeatherData.jar
«Artifact» gson.jar
«StormServerControlPa Component
<<WritableGUI>>
Java Sockets
TCP/IP Ethernet
<<StormPi Sensor>> OS:Rasbian
<<execution Environmenrt>> : JVM
«Temperature.java» Component
«Pressure» Component
«StormPIManager.java» Component
<<StormBoard>>
<<uses>>
«Register.java» Component
<<WritableGUI>>
«StormPI» Component

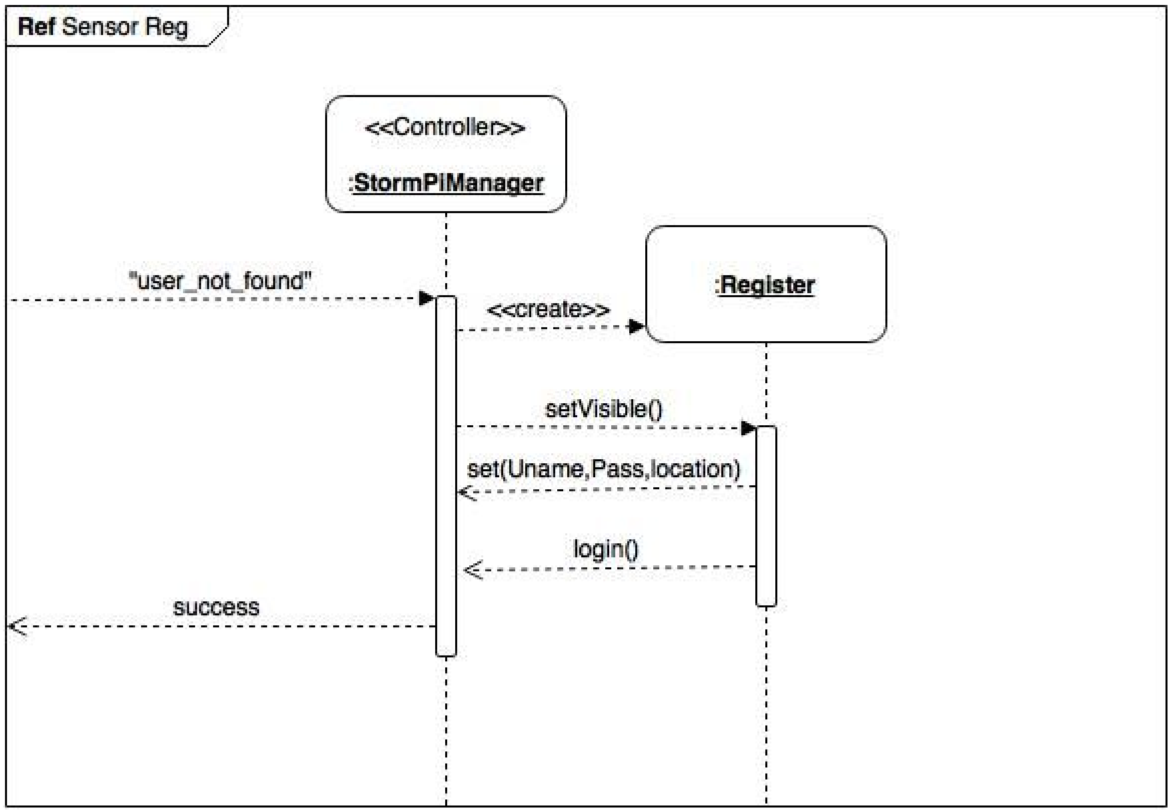
Sequence diagram[1] - StormPI Weather Sensor



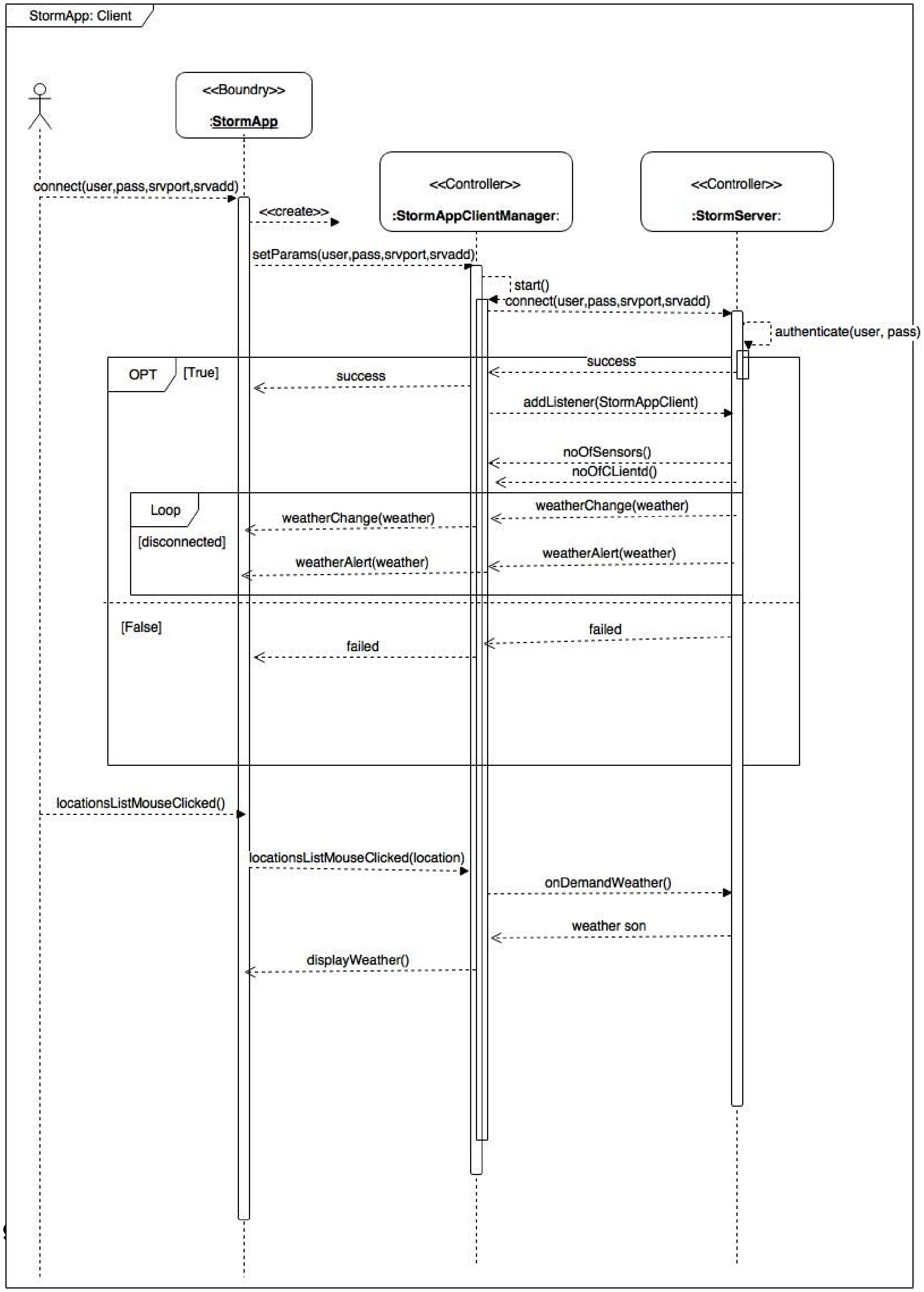
Sequence diagram - StormPI Sensor: Connecting to Storm Server



Sequence diagram - Sensor Registration (Associated with “Connecting to Storm PI diagram” given above)



Sequence diagram - RMI StormAPP and StormServer Communication



Requirements

Storm can be deployed on any hardware device which supports the following software requirements.

1. Mac OSX
2. Windows 10
3. Java 1.8
4. Ubuntu 16.4

Assumptions

- StormPI which used to simulate a real world sensor is nearly identical to a real system.
- On demand weather data of the StormAPP client(monitor) outputs the latest weather update of that particular location, not the history of weather data.
- Using a file to store registration won't affect the efficiency of the server.
- Garbage collector will automatically destroy all daemon threads, if exist any.

Known Issues.

- Client throws null pointer exceptions when running on fedora linux and ubuntu 16.4

References

[1] "Object_Exportation", http://www.fitc.unc.edu.ar/javadev/rmi/sequence_diagrams.html#callExec
[2] "Deployment Diagrams Overview",
"http://www.uml-diagrams.org/deployment-diagrams-overview.html"

Extra

All the diagrams included in this report are provided in the diagrams folder for a better view.