Software Requirements Specifications

1. Introduction
   - 1.1. Authorship
     - 1.1.1. Peter Banis
     - 1.1.2. Klaus Cipi
     - 1.1.3. Michael Kolar
     - 1.1.4. Robert Olsen
   - 1.2. Purpose

     This SRS shall define the obligations, characteristics, features and constraints of the ad-hoc wireless networking project. This shall regard the API itself and a tool to verify the installation of the API. This document is intended for Dr. Silaghi, the client of our team, as well as any stakeholders who would require access to determine the status of the project, provide input to, or evaluate the project. This may include but is not limited to Dr. Chan (organizer of senior design projects for the Fall 2018 - Spring 2019 year), judges at a showcase, or anyone who may wish make use of the results of our work.
   - 1.3. Scope

     The API and Installation Verification Tool will be compatible with Windows 7, Windows 10, MacOS, Android, and Linux. It shall enable communication from Android's P2P Standard and the existing ad-hoc infrastructure supported by Windows, MacOS and Android.
   - 1.4. Definitions and Acronyms
     - 1.4.1. IEEE = Institute of Electrical and Electronics Engineers
     - 1.4.2. API = Application Programming Interface
     - 1.4.3. P2P = Peer-to-peer (wireless network)
     - 1.4.4. OS = Operating System
     - 1.4.5. SRS = Software Requirements Specifications
     - 1.4.6. URL = Uniform Resource Locator (web address)
     - 1.4.7. JVM = Java Virtual Machine
   - 1.5. References
     - 1.5.1. This document shall adhere to the following publications:
       - 1.5.1.1. IEEE 830-1998, IEEE Recommended Practice for Software Requirements Specifications
       - 1.5.1.2. IEEE Wi-Fi Peer-to-Peer (P2P) Technical Specification Version 1.7
       - 1.5.1.3. IEEE 802.11
   - 1.6. Overview

     The SRS contains all relevant information for stating the requirements of the project. It shall define what is required of the system for functionality, correctness, and validation. It shall further describe the purpose of the system. This may include those who would use the system directly - software engineers, for example - and those who would use the system indirectly, such as users of mobile applications. The document is organized into the following sections: the overall description which contains information about the user interfaces, hardware interfaces, software interfaces, requirements for the API and requirements for the installation verification tool, expectations regarding both direct and indirect users of the system (i.e. engineer and user using that engineer's work), as well as any constraints that must be placed on the system and any assumptions made about the system, and the devices using the system or any dependencies that it will require to function. The document will then describe the special requirements that include but are not limited to: performance, ease of use, documentation, and extensibility.

2. Overall Description

2.1. Product Perspective

2.2. The API is self contained and is not a component of another system.

2.2.1. User Interfaces

2.2.1.1. The installation verification tool shall report to the user the following information: any file in the API that does not exist in the users directory, or any file in the API that has incorrect permissions for its function.

2.2.2. Hardware Interfaces

2.2.2.1. On Android devices, the API will require the hardware to support the P2P standard.

2.2.2.2. The API shall be compatible with older Android models.

2.2.3. Software Interfaces

2.2.3.1. The API shall require the existence of BASH on Linux and MacOS systems.

2.2.4. Communications Interfaces

2.2.4.1. The API shall communicate over UDP and TCP/IP network protocols

2.2.5. Memory

2.2.6. Operations

2.3. Product Functions

2.3.1. The Java API shall have a unique set of functions:

2.3.1.1. The API shall support MacOS, Linux, Android, and Windows (7/10 at least).

2.3.1.2. The API shall be able to detect whether a P2P or ad-hoc connection is possible and clarify this to the user.

2.3.1.3. The API shall enable communication across Android's P2P standard and the existing ad-hoc system whenever possible. (See Fig 1, 2)

2.3.1.3.1. If the communication is not possible, an exception shall be raised.

2.3.1.4. The API shall allow the user to choose which type of connection to make.

2.3.1.5. The API shall allow connections between any combination of Android, Linux, MacOS, and Windows systems.

- 2.3.2. The installation verification tool shall have its own set of functions:
  - 2.3.2.1. It shall check to make sure that all necessary files are present in the file system.
  - 2.3.2.2. It shall check to make sure that any scripts have the correct permissions.
  - 2.3.2.3. If any files are missing, then the installation verification tool will tell the user what files are missing.
  - 2.3.2.4. If any files have incorrect permissions, then the installation verification tool will tell the user which files have discrepancies and what the permissions should be.
  - 2.3.2.5. There shall exist a tool component to correct permissions for all files included in the API.
  - 2.3.2.6. There shall exist a tool component to determine if the host Android device is capable of using the P2P connection at all.

2.4. User Characteristics

- 2.4.1. Programmers
  - 2.4.1.1. These programmers are expected to have a high level of technical expertise.
  - 2.4.1.2. They will have an interest in network programming or applications that utilize network programming.
  - 2.4.1.3. Their general educational level isn’t very relevant, but it can be expected that many will have at least a Bachelor's degree in Computer Science.
  - 2.4.1.4. They will likely be working on application development which requires cross-platform capabilities.
  - 2.4.1.5. They will likely be interested in smartphone applications.
  - 2.4.1.6. They will have quality expectations for the API:
    - 2.4.1.6.1. They will expect that all methods added to the API have been tested thoroughly. This will allow them to reasonably conclude that any errors in their applications are of their own making and not an error in the API.
    - 2.4.1.6.2. They will want method names which are easy to remember and concisely communicate the function of the method.
    - 2.4.1.6.3. They will want proper documentation which will specify the valid inputs and expected outputs of each method.
- 2.4.2. Indirectly, people who use any form of P2P or ad-hoc networking will also be users.
  - 2.4.2.1. This will include essentially anyone with a device capable of connecting to the internet.
  - 2.4.2.2. They will have a wide range of technical expertise from novice to expert. However, their technical expertise is

circumstantial since any applications that use the API should be tailored to work without tinkering by these users.

2.5. Constraints

2.5.1. The API shall handle all handshake protocols required by the 802.11 or P2P standards

2.5.2. The API shall require hardware support for its actions

2.5.2.1. Android devices shall have P2P capabilities

2.6. Assumptions and Dependencies

2.6.1. The API shall not make any assumptions about how it is to be used without explicit statement. That is, if the API assumes a function A will be used preceding a function B to ensure proper behavior, the documentation of function B shall make a clear and unambiguous statement declaring this.

2.6.2. The installation verification tool shall assume it is installed with the API in the same directory.

2.6.3. The API will have a hardware dependency on Android devices to support the P2P protocol. The API shall not assume the Android device does, but shall inform the user of the system when the device is not supported.

3. Specific Requirements

3.1. External Interface Requirements

3.1.1. User Interfaces

a) Installation validation.
b) Detailed information related to the kind of error that occured at a specific time and place provided to the developer to easily debug the issue.
c) Depending on how the API is installed, feedback will be displayed in the terminal or the IDE's terminal.
d) Feedback will be accurate up to the file that the problem occured in or for files that are missing, but that does insure that in rare cases the problem might not have occured in the provided file.
e) Units of measure do not apply.
f) Feedback will be provided within 1ms after the issue is detected.
g) Does not relate to any other input or output.
j) Files : string containing their location will be provided.
Date and Time : string containing current date and time in UTC format will be provided.
Progress and Possible solutions : a string containing a series of instructions will be provided.
h - k) Format:

```
Installation failed!
[Date and Time]
[Corrupted Files (if any)]
```

[Missing Files (if any)]
[Progress done (up until the issue)]
[Possible solutions]

3.1.2. Hardware Interfaces

a) P2P or ad-hoc connection
b) The API shall send a request to the device to make sure that P2P is supported. If P2P connection is not available, a standard ad-hoc connection will take place.
c) Device response shall be provided as input for the API.
d) The API shall either start a P2P or ad-hoc connection, and it shall start a connected over 96% of the time.
e) Seconds.
f) A connection shall start in 5s, and shall have a 25s time-out.
g) If the API fails to start a connection, a warning will be displayed to the user.
h - i) N/A
j) String
k) Start Connection: string

3.1.3. Software Interfaces

a) BASH software
b) The API shall be provided with BASH software on MacOS and Linux
c) No input/output
d) BASH shall be installed on MacOS and Linux
e) N/A
f) N/A
g) BASH shall be installed for the API to be installed
h - l) N/A

3.1.4. Communications Interfaces

a) UDP standard
b) Client-server communication shall be acquired through UDP broadcasting.
c) Input shall be information that shall be send and the output shall be a UDP datagram.
d) The API shall complete 99% of all communications.
e) Packets (Datagrams)
f) Timing will depend on the size of the packets.
g) The API shall retrieve information from the application and it shall output results to the application.
h-i) N/A
j) Datagrams

3.2. System Features

3.2.1. Automatic OS Detection

3.2.1.1. Introduction/Purpose of Feature

A single Java API package shall include support for all major platforms (Android, Windows, Linux, and MacOS). The API shall automatically detect the platform that the developer is working on and provide the appropriate support for it. Furthermore, during the installation of any application that will use this API, only scripts for the proper platform shall run.

3.2.1.2. Stimulus/Response Sequence

The API shall invoke automatic OS detection function (stimulus), which shall return the OS name (response) of the device that the API is being installed on.

3.2.1.3. Associated Functional Requirements

3.2.1.3.1. The API shall provide information about the OS.

3.2.2. Integrity Check

3.2.2.1. Introduction/Purpose of Feature

The system shall ensure that all components have been installed correctly. This is not limited to only ensuring the correct files exist in the correct directories, but also that all files have needed permissions (for example, a script might lack an execute permission which prevents further progress). The system shall recognize when such a situation occurs, and if it cannot be fixed by the system alone, the system shall provide useful feedback to the user so they may correct the error.

3.2.2.2. Stimulus/Response Sequence

Whenever an exception is caught (stimulus), the API shall provide meaningful response to the developer (response).

3.2.2.3. Associated Functional Requirements

3.2.2.3.1. The API shall catch exceptions

3.2.2.3.2. The API shall provide possible solutions

3.2.2.3.2.1. URL for missing files

3.2.2.3.2.2. Other suggestions

3.2.2.3.3. The API shall check for missing files

3.2.2.3.4. The API shall check for corrupted files

3.2.3. Utilizing Ad-hoc Connection

3.2.3.1. Introduction/Purpose of Feature

The API shall take advantage of Android's native hardware capability for simultaneous dedicated internet connection and ad-hoc connection.

3.2.3.2. Stimulus/Response Sequence

Whenever an application that is build upon this API requests to open an ad-hoc connection (stimulus), the API shall open the connection (response).

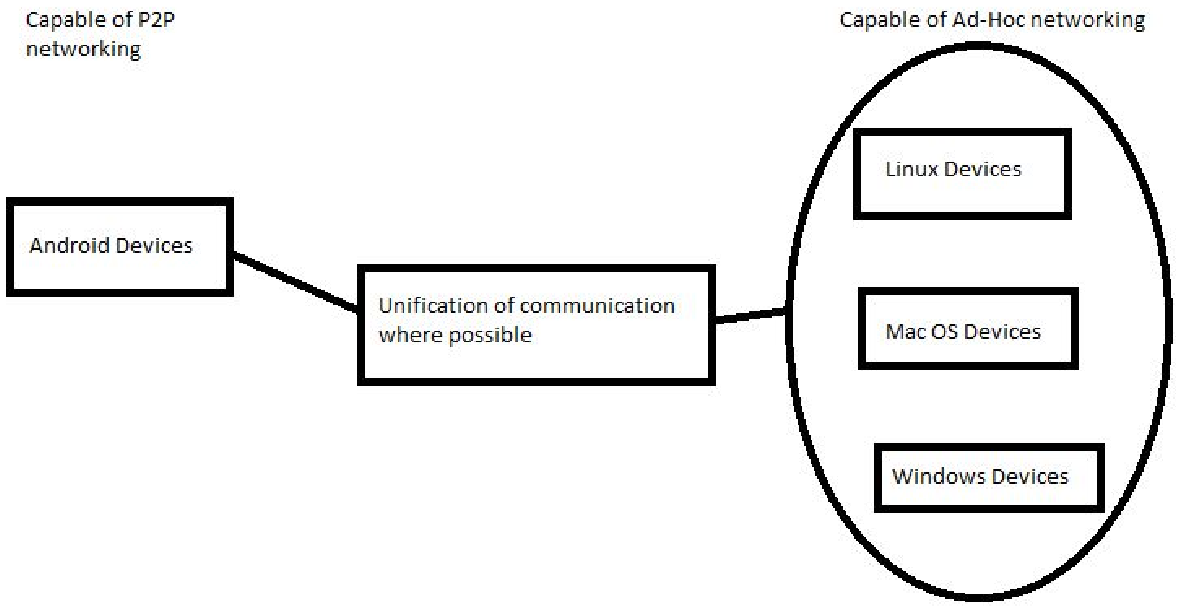
        3.2.3.3. Associated Functional Requirements
            3.2.3.3.1. The API shall test if the device is capable of opening an ad-hoc connection.
            3.2.3.3.2. The API shall open ad-hoc connection.

3.3. Performance Requirements
    3.3.1. A P2P or ad-hoc connection shall start within the time period specified by the 802.11 standard or P2P standard as appropriate.
    3.3.2. A connection session shall terminate after 5 minutes of inactivity or any limit set by the 802.11 or P2P standard when appropriate.
    3.3.3. File or information transfer shall start within 1s after application’s request.
    3.3.4. A file or information transfer shall terminate after 1 minute of connection loss.

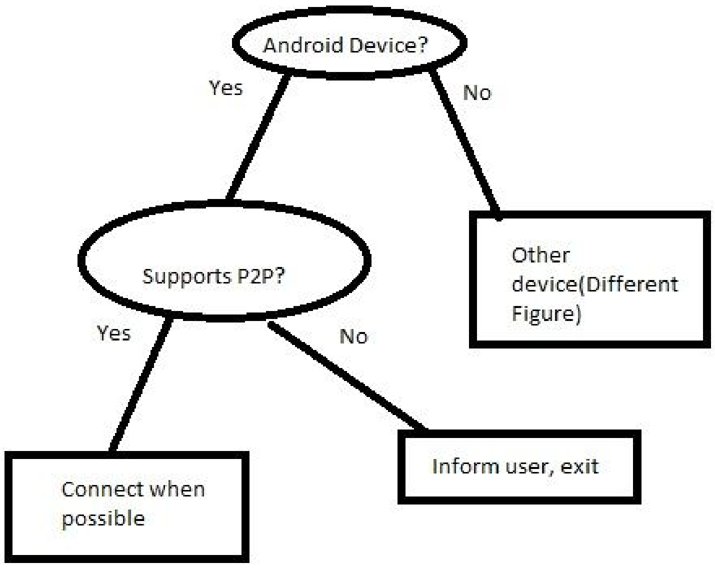
3.4. Design Constraints
    3.4.1. Connection shall be P2P if the hardware supports it.
    3.4.2. The API shall be installed on MacOS 10.5 or later, Windows 7 or later, Linux, or Android 4 or later.
    3.4.3. MacOS and Linux shall have BASH installed.
    3.4.4. The device shall support at least ad-hoc connections.
    3.4.5. The host device shall be able to install JRE or JDK.

3.5. Software System Attributes
    3.5.1. Security
        3.5.1.1. The API shall not implement any encryption in communication so that the application built on top of the API shall provide its own encryption.
        3.5.1.2. Developers shall not have direct access to variables.
    3.5.2. Maintainability
        3.5.2.1. The API shall have detailed documentation.
    3.5.3. Portability.
        3.5.3.1. The API shall be written in Java.
        3.5.3.2. The API shall be compiled on JVM.

Appendices


Capable of P2P networking
Capable of Ad-Hoc networking
Android Devices
Unification of communication where possible
Linux Devices
Mac OS Devices
Windows Devices

(Fig 1) (Ad-Hoc / P2P communication)



(Fig 2) (P2P communication)

Table of Contents