Project Title
A Java API for unifying ad-hoc Wifi networking

Team Members
Klaus Cipi - kcipi2015@my.fit.edu
Peter Banis - pbanis2015@my.fit.edu
Michael Kolar - mkolar2015@my.fit.edu
Robert Olsen - olsenr2015@my.fit.edu

Faculty Sponsor
Dr. Marius Silaghi - msilaghi@fit.edu

Client
Dr. Marius Silaghi - Associate Professor at College of Engineering & Science, FIT

Meetings with Sponsor/Client
Monday, September 17, 2018

Progress of current Milestone (progress matrix)

Task	To Do (Completion %)	Peter	Klaus	Michael	Robert
Decide between developing on MacOS, Windows, or Linux	None (100% Complete)	Discussion	Discussion	Discussion	Discussion
Decide whether to use Eclipse or IntelliJ as an IDE	None (100% Complete)	Discussion	Discussion	Discussion	Discussion
Install and configure Android Studio	None (100% Complete)	Do	Do	Do	Do
Install and connect to the SVN repository provided by Dr. Silaghi	None (100% Complete)	Do	Do	Do	Do
Successfully install the API	None (100% Complete)	Linux	MacOS	Windows	Android
Run an example program with the API	None (100% Complete)	Do	Do	Do	Do

Create Requirements Document	None (100% Complete)	20%	50%	20%	10%
Create Design Document	None (100% Complete)	10%	50%	20%	20%
Create Test Plan	None (100% Complete)	90%	0%	10%	0%

Discussion (at least a few sentences, ie a paragraph) of each accomplished task (and obstacles) for the current Milestone

- Decide between developing on MacOS, Windows, or Linux: The consensus was Windows. We were strongly pulled towards Windows due to the commonality of the operating system in our devices; most of our team members have devices where Windows is the default operating system. Our only obstacle here is that one group member doesn't have access to a native Windows device; however, this can be circumvented through the use of dual-booting. Windows required the minimum adjustment for our group.
- Install and configure Android Studio: This was a basic task that didn't provide any notable obstacles. Android Studio will be a valuable tool throughout the project because it is designed specifically for the development of Android development. This is relevant because the point of the project is to finish the API for the Android operating system.
- Decide whether to use Eclipse or IntelliJ for an IDE: We decided on IntelliJ. IntelliJ stands over Eclipse when it comes to convenience of use. Most of this stems from IntelliJ's understanding of context. In Eclipse, in order evaluate an expression the user must highlight the entire expression, but IntelliJ is much more aware of where the user's mouse is so it only requires that the user hover over the expression. It's a small thing, but it speeds up the debugging process. IntelliJ also has superior auto-complete. In Eclipse if a word is left unfinished it will display every possible name that could follow. However, IntelliJ, to an extent, can determine what data type is needed for parameters and locate static variables which satisfy the text and type requirement. This makes its auto-complete feature far more accurate. IntelliJ also offers a feature where it will offer variable names that sound sensible to a human programmer; it uses every relevant method name, variable type, and even values to determine a suitable name. This helps prevent name collisions because IntelliJ checks for duplicates.
- Install and connect to the SVN repository provided by Dr. Silaghi: Everyone installed the SVN repository by using the TortoiseSVN application. We used the Checkout feature

which allowed TortoiseSVN to create a working copy of the repository provided by Dr. Silaghi via the URL. These copies are stored locally on our computers.

- Successfully install the API: This ties into “Install and connect to the SVN repository provided by Dr. Silaghi”. The repository he provided contained the API along scripting files
- Run an example program with the API (to verify correctness of install and behavior): The repository came with a file called “test.java” which would test the classes already in there. We ran that file and the output suggested that the API is working properly. However, in the future as we start to develop our own classes we are going to need to write our own test programs.
- Create Requirements Document: The requirements document was completed. Confusion regarding the implementation of ad-hoc with respect to the pre-established P2P system, as well as the extent of backward compatibility with MacOS and Android. A few responsibilities of the operating system were mistakenly attributed to the API itself, although this has been corrected.
- Create Design Document: The design document was completed. One of the obstacles of writing it was that the format required it to be written in how the API would affect various stakeholders. An API is a niche tool that will only be directly used by other programmers, so at first it wasn’t clear which parties would be relevant stakeholders. However, the group’s thought process adapted and branched out into parties that are indirectly influenced by the API.
- Create Test Plan: The test plan document was completed. There were not any noticeable problems with creating the plan itself, however our limited knowledge of the P2P standard and the somewhat exploratory nature of the bridge between Ad-Hoc and P2P devices made it difficult to devise a proper test method. The other components of the API and verification tool were much more straightforward in their purpose and easily lend themselves to a “test->result” format.

Discussion (at least a few sentences, ie a paragraph) of contribution of each team member to the current Milestone

- Peter Banis: Peter essentially single handedly wrote the test document. He also provided a substantial portion of the requirements document, and contributed to the design document. He took an active role in keeping the group on track and determining what needed to be done. He successfully completed the individual milestones: install IntelliJ, connect to the SVN repository, and use TortoiseSVN to download the starting API.
- Klaus Cipi: Klaus wrote the majority of the design document, and at least half the requirements documentation. He successfully completed the individual milestones:

install IntelliJ, connect to the SVN repository, and use TortoiseSVN to download the starting API.

- Michael Kolar: Michael contributed to the requirements document, design document, test document, progress evaluation form, and presentation for Milestone 1. He successfully completed the individual milestones: install IntelliJ, connect to the SVN repository, and use TortoiseSVN to download the starting API.
- Robert Olsen: Robert contributed to the requirements document, design document, and created the presentation for Milestone 1. He successfully completed the individual milestones: install IntelliJ, connect to the SVN repository, and use TortoiseSVN to download the starting API.

Plan for the next Milestone (task matrix)

Task	Peter	Michael	Klaus	Robert
Integrity check and installation validation (IT&D)	25%	25%	25%	25%
Automatic OS detection (IT&D)	25%	25%	25%	25%

Discussion (at least a few sentences, ie a paragraph) of each planned task for the next Milestone

- Integrity check and installation validation (IT&D): This will be a new java file (to maintain platform independence) which will search the existing directory for each script and java file provided with the API. This file will both check the existence of the files and report to users any missing files. Additionally, the tool will be able check the Read/Write/Execute permissions on all files, and inform users of files which lack needed permissions. Finally, the tool will be able to detect if an Android device is capable of P2P networking and report this to users.
- Automatic OS detection (IT&D): This relates to various capabilities. We don't intend to begin Android P2P development and the Ad-Hoc bridge yet, however this will be important for determining what features of the API can be enabled(if it turns out that Android devices can only use a subset of the Ad-Hoc standard). This also relates to the above task. There is no reason to require the API on windows to require .sh scripts for Mac/Linux systems, so we can reduce a burden on some users by requiring less for them.

Sponsor feedback on each task for the current Milestone

- Decide between developing on MacOS, Windows, or Linux:

- Install and configure Android Studio:

- Decide whether to use Eclipse or IntelliJ for an IDE:

- Install and connect to the SVN repository provided by Dr. Silaghi:

- Successfully install the API:

- Run an example program with the API (to verify correctness of install and behavior):

- Create Requirements Document:

- Create Design Document:

- Create Test Plan:

Sponsor Signature: _____ Date: _____

Sponsor Evaluation

- Sponsor: detach and return this page to Dr. Chan (HC 322)
- Score (0-10) for each member: circle a score (or circle two adjacent scores for .25 or write down a real number between 0 and 10)

Peter Banis	0	1	2	3	4	5	5.5	6	6.5	7	7.5	8	8.5	9	9.5	10
Klaus Cipi	0	1	2	3	4	5	5.5	6	6.5	7	7.5	8	8.5	9	9.5	10
Michael Kolar	0	1	2	3	4	5	5.5	6	6.5	7	7.5	8	8.5	9	9.5	10
Robert Olsen	0	1	2	3	4	5	5.5	6	6.5	7	7.5	8	8.5	9	9.5	10

- Sponsor Signature: _____ Date: _____