**Week 8: Analyze an Open Source Project**

Nate Bachmeier

April 28th, 2019

TIM-8101: Principals of Computer Science

Northcentral University

# Bottoms up Analysis using F-Droid

F-Droid is an open source mobile application store for the Android platform. It allows developers to share source code and an easy model for side loading compiled versions of the applications onto physical devices.

Normally open source repositories are analyzed from the top-down, however this assumes one even knows what is relevant to the final binary. Instead a macro bottoms up solution was performed across three passes. The first pass downloaded 1610 APK files and extracted high level features. Next decomplication was performed on 339 files, and finally specific interesting characteristics was explored in one of them.

# Finding Interesting Project on F-Droid

## Downloading APK Files

F-Droid exposes a complete directory of projects hosted by their platform. Each directory entry has a standardized template which was parsed to get the source code and the compiled Android PacKage (APK). When multiple versions were available, only the most recent was selected for download resulting in 1610 results totaling approximately 6.5gb of binary files.

The median file size was 4.2mb with a standard deviation of 7.8mb and a maximum size of 88.9mb. The ignore the effects of the long tail the 99th percentile was used as the as largest size resulting in 39.8mb.

APK files use the same format as regular ZIP archives and can be extracted with commodity decompression software. The root cause for many of the large APK is too many resource files. Some APKs, like com.dkanada.icecons.apk, include multiple resolutions of the graphics and then select at runtime. Others mitigated this issue by choosing at build time and creating one APK per device category (eg. SmartTV vs mobile phone).

## Opening the APK

Inside of each archive is AndroidManifest.xml, classes.dex, lib folders, and res folders. The manifest describes the permissions and public symbols of the application. Dalvick EXecutable files is compiled code for the virtual machine. The LIBraries folder contains any platform native binaries that need to be consumed through the Java Native Interface. RESources are images, configuration, and layout templates.

Using the dex2jar-2.0 and OpenJDK’s javap utilities a randomly sampled 25% subset of classes.dex files were transformed first into Java Archive format. The archives were expanded to get the Java Class files, which were then converted into Java Assembly files. The Assembly files were then mined to extract a graph representation of the inheritance and call trees. The graphs were then persisted as GraphML files which can be imported into existing graph analysis software.

## Common Packages

Roughly 65% of sampled application are linked against Android Jetpack. Jetpack is removes common boiler plate code and accelerates the time to develop new applications.