**MSC Advance Software Engineering – Thesis project proposal**

**Mick O’Doherty - 12259095**

**Performance of locally connected collaborative devices**

# Overview

This project will explore how a set of locally connected computing devices can collaborate to improve the performance of applications running on a given device within the set.

In particular the following proposition will be tested:

The real world performance, and the perceived performance, of CPU intensive applications, such as video transforming applications, running on personal computing devices can be improved by breaking down and sharing certain tasks with other associated personal computing devices that are in close proximity, when those other devices have spare processing capacity.

# Background

It is quite often the case, with the recent growth of computing power in mobile devices, that a user will have multiple relatively powerful computing devices in close proximity at any given time.

For example a mobile worker will generally have at least a mobile phone and a laptop when working remotely. Some may have a tablet in addition to this and it is not unusual for workers to travel with more than one mobile phone or tablet.

For certain compute intensive tasks it may be the case that the user is frustrated by the performance of an application which has a high demand for processing power on the current device they are using, while their other devices are sitting relatively idle with plenty of spare computing capacity.

This project aims to explore whether compute intensive tasks on one device can improve their performance by offloading some of their work load to other associated devices in the immediate vicinity.

In some respects this is analogous to a local computer offloading work to ‘the cloud’, or a server in a data center distributing its load to others servers in the same data center – however, in this case the cloud or servers are a small local group, or ‘cloud’, of associated devices in close proximity to the local computer.

# Approach

An Android tablet will be used as the primary device on which the performance of a set group of applications will be measured. The test will feature a repeatable script and will be run multiple times to gather average timings for each test scenario.

Two main mechanisms will be developed for the Android tablet to share CPU intensive tasks with other associated local devices:

* A mechanism to distribute suitable Android Services running on the tablet to other devices in the vicinity
* A mechanism to break down large CPU intensive workloads into chunks which can be implemented in parallel at differing speeds on non-heterogeneous devices in the vicinity

The tests will run a controlled set of applications in a tablet that uses the collaborative workload sharing techniques above and one that does not, and the performance results will be compared over multiple test runs.

For the purposes of this project the associations between the devices will be statically defined on each device, and the focus will be on working with large video files, as this is a common high CPU load task.

# Output

The project will produce a number of outputs:

* A mechanism to distribute Android services to associated devices.
* A mechanism to break down a video transformation task on an Android device, share it across associated devices and combine the output back on the original device
* A documented test script and collection of tests cases to measure the performance of a given set of applications on an Android Tablet.
* A report containing the results of running the above test suites on a standard Android tablet and on one that uses the task sharing mechanisms above.
* A summary, including a conclusion which either supports or calls into question the performance improvement premises of the approach explored in this project.

# Work to date

* Investigated getting ffmpeg executable running on Android and calling from Android app via exec command.
* Rejected due to issues with exec command in Java and need for rooted devices.
* Use Android NDK to create ffmpeg library and call code programmatically via JNI interface from Android application.
* Getting ffmpeg running in an App via JNI proved extremely complicated with multiple issues even with commonly available instructions when using a MAC. One additional output of this project will be a clear set of instructions for creating an Android ffmpeg application using Eclipse on an OSX generation MAC.
* Basic ffmpeg Android App test mule working on Emulator as of September 2014.

# Updated Plan after initial investigation

## Phase 1 - Basic App

* User records a video on their phone or else has a video preloaded on the phone
* User starts. app and selects video and selects soundtrack, which is also located on their phone.
* App merges video and soundtrack.
* User plays video.

## Phase 2 - Distributed operations

* User records a video on their phone or else has a video preloaded on the phone.
* User starts app and selects video and selects soundtrack, which is also located on their phone.
* App splits video into 4 equal parts.
* App sends video to all companion devices and also merges audio and video for its part.
* App retrieves parts from companion devices.
* App builds complete video from completed video.
* User plays video.

## Phase 3 (stretch) - Dynamic distributed

* User records a video on their phone or else has a video preloaded on the phone.
* User starts app and selects video and selects soundtrack, which is also located on their phone.
* App detects how many companion devices are available to help.
* App splits video into equal parts depending on number of companion devices.
* App sends video to all companion devices and also merges audio and video for its part.
* App retrieves parts from companion devices.
* App builds complete video from completed video parts.
* User plays video

## Testing

### Measurements

#### Elapsed time:

* Measure total elapsed time to merge video and audio on single device with very large video
* Measure total time to merge video and audio on main device plus 3 companion devices with same very large video

#### CPU time:

* Measure total CPU time on main device to merge video and audio on single device with very large video
* Measure total CPU time on main device time to merge video and audio on main device plus 3 companion devices with same very large video

#### Battery life:

* Find sensible way of measuring battery usage if possible
* Test with single app
* Test with companion app

# Annex - Original Phasing for reference

## Part 1 – Baseline

This phase delivers a working implementation of a system that splits a pre- defined video manipulation task into a number of sub-tasks which are then executed in parallel on multiple Android devices.

## Part 2 – Self management

Part 2 builds on Part 1 and adds group self management capability to the system. This capability will allow the system react to the addition and removal of worker nodes from the group without having to restart the overall system task. It will also include the ability to send more work to any node in the group which completes early.

## Part 3 – Dynamic collaboration

Part 3 will develop protocols for node capability advertisement and discovery. These protocols will allow Android devices collaborate to distribute and share workload in adhoc and dynamic work groups.