CSE 535: Mobile Offloading Project - Final Report

Eshan Gaur
Arizona State University
Tempe, AZ, US
egaur@asu.edu

Mayank Batra
Arizona State University
Tempe, AZ, US
mbatra3@asu.edu

Quinn Fischer
Arizona State University
Tempe, AZ, US
qfischer@asu.edu

William Gibbs
Arizona State University
Tempe, AZ, US
wfgibbs@asu.edu

ABSTRACT

A Peer-To-Peer Android mobile offloading application. The application establishes connection between nearby devices through local WiFi communication. The user must then allow application to share location and battery data and agree to offer computational resources for another user on the network. Application prioritizes nearby and high battery devices and then partitions the matrix multiplication problem between devices. The master/requesting device pieces together the resulting sub matrices and thus answers a matrix multiplication problem through mobile offloading.

KEYWORDS

Mobile Offloading, Local WiFi, Block Matrix Multiplication, Android, Peer-To-Peer (P2P)

INTRODUCTION

Modern technology in the form of smartphones and mobile devices allow computational resources to be almost anywhere. Yet, this mobility come at a cost of being power/battery restricted and consequentially less powerful than stationary computational infrastructure. To address these constraints mobile offloading is proposed, yet instead of relying upon fog or cloud infrastructure, which could be compromised or unavailable in remote or disaster settings, we look into connecting mobile devices together to form a network [1]. By putting mobile resources together and partitioning computational problems, we can achieve better performance through the power of concurrent computing. This allows mobile users to come together even in disaster and internet-free settings to coordinate and solve complex computational problems given the resource constraints of mobile devices.

PROJECT ARCHITECTURE

Configurations

This mobile application is designed in the style of client-server or master-slave like architecture. Except, one device is chosen to behave as the Master and the other devices aid in computation as slaves. The Project setup involves installation of the signed application on the device, which runs Android with following configurations:

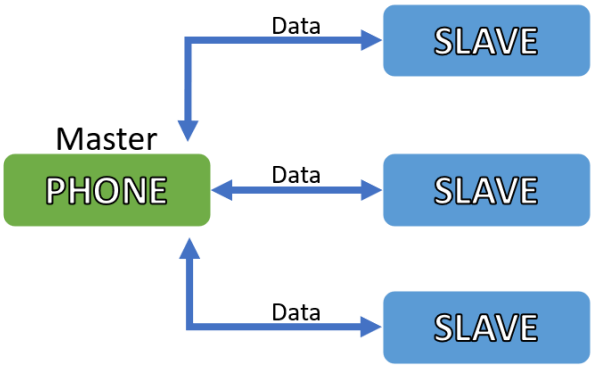

Figure 1: Diagram of master-slave architecture employed by application

1. Compile Sdk Version 29
2. Build Tools Version "29.0.3"
3. Minimum Sdk Version 21
4. Target Sdk Version 29

Permissions

The Application also runs with the following permissions enabled:

1. Permission name = ACCESS_WIFI_STATE
2. Permission name = CHANGE_WIFI_STATE
3. Permission name = ACCESS_FINE_LOCATION
4. Permission name = ACCESS_COARSE_LOCATION
5. Permission name = CHANGE_NETWORK_STATE
6. Permission name = INTERNET

7. Permission name = ACCESS_NETWORK_STATE
8. Permission name = WRITE_EXTERNAL_STORAGE
9. Permission name = WRITE_INTERNAL_STORAGE

Setup Scenario

The following is an example scenario of how to use the application:

1. Install the application
2. Wait for the application to launch
3. Once the application is launched, enable the device's WiFi
4. Then click the "Extract Data" and allow the application access to location services via the popup window that appears
5. Ensure all devices to be networked together are at this step then continue
6. Next, press the button to discover nearby devices
7. Then, nearby devices will display on the application in the list view
8. Select the relevant device from list view. This will prompt a connection request on the selected device
9. Accept the request on the selected device and the network is established
10. Location and battery information is now being collected. User can see a sample of the data being collected by clicking the "Extract Data" button followed by pressing the "Transfer Data" button on a device. The sampled information is then displayed on the other device
11. On the master application two matrices can be provided for offloading in the provided edit text views (In Java like string matrix format)
12. The distributed matrix calculation can then be began by pressing the "Offload Matrix Calculation" button
13. The master then proceeds to divide the task into sub matrices, sending each to a contributor device, which then solves the sub problem and returns it to the master device
14. Once all sub matrices are received by the master device, it then pieces the problem together into the full matrix problem and displays the result to the display
15. Furthermore, Estimations of the power usage, execution time, and failure response time can then be seen by pressing the "Calculations" button

IMPLEMENTATION

We now discuss our project's implementation by partitioning the requirements based on their underlying purpose:

1. *Developing the Application to connect the phones together in a network.* The application needs to be able to establish a connection [2] between phones with the application over local WiFi bands [3]. To do so the user must first open the application, enable WiFi access on the phone, and enable location services for the application [4].

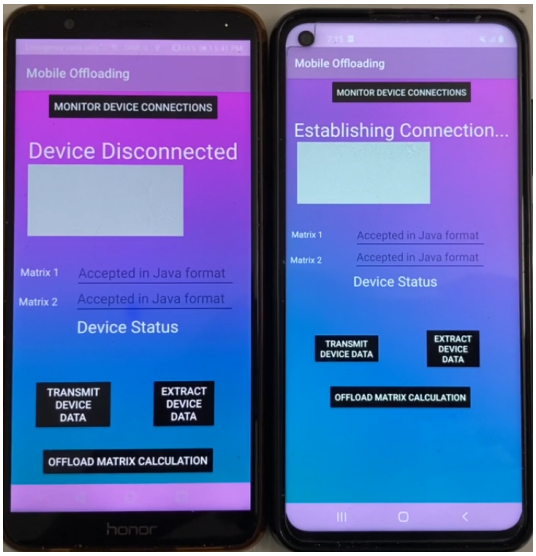

Figure 2: Two phones with application open, side-by-side

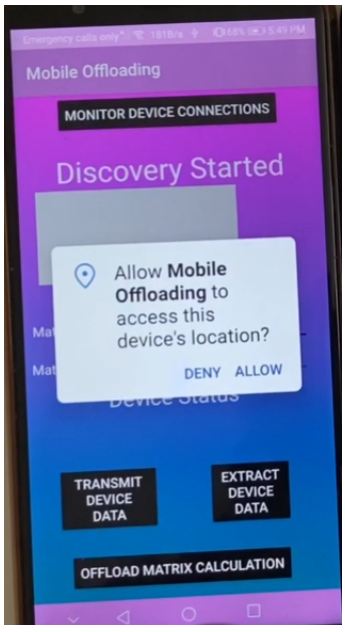

Figure 3: Allowing access to device's location data

2. *Connecting the mobile devices together.* Once WiFi and location services are enabled [5], the user can

then select the device he wishes [6] to connect with using WiFi Direct protocols.

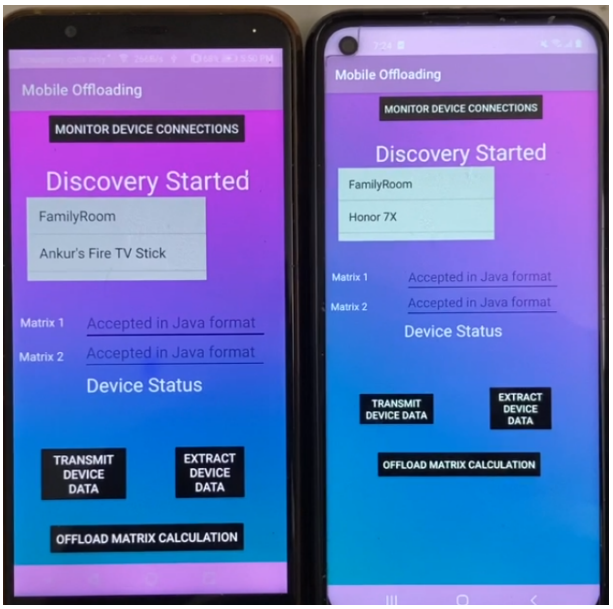

Figure 4: Nearby accessible WiFi devices are listed

3. *Allowing the mobile devices to connect together in a network.* Once the user accepts, the one who initialized the request is made the Master and everyone else are made participants.

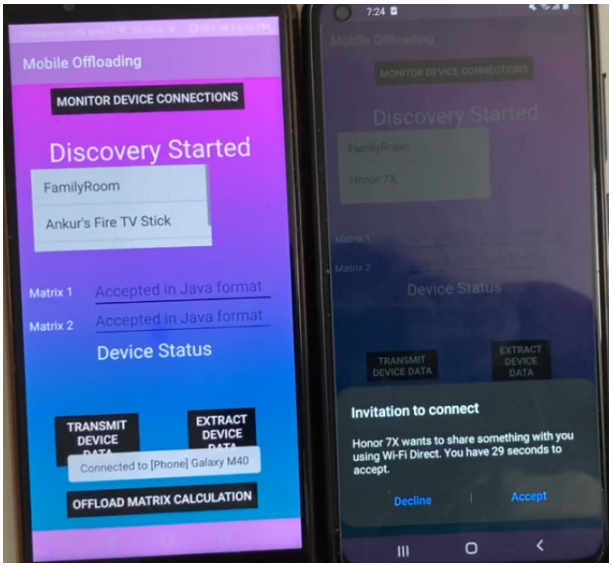

Figure 5: Sending a connection request query between the devices

4. *Requesting battery and location information between master and participants.* Once the connection network is established, the Master application can then request location [7] and battery information [8] from the participants/client applications.

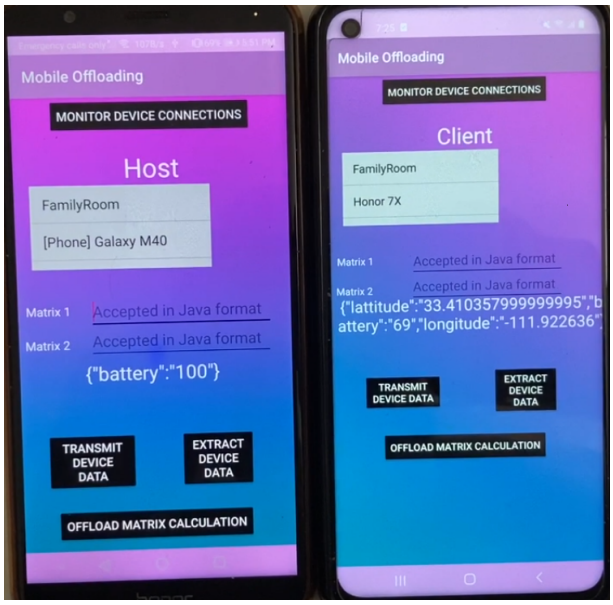

Figure 6: Battery and location data is shared between the devices

5. *Initializing matrix multiplication.* Then the matrix multiplication task can be initialized on the Master device. In our application, the user enters the matrix strings according to Java specifications that the user wishes to calculate.

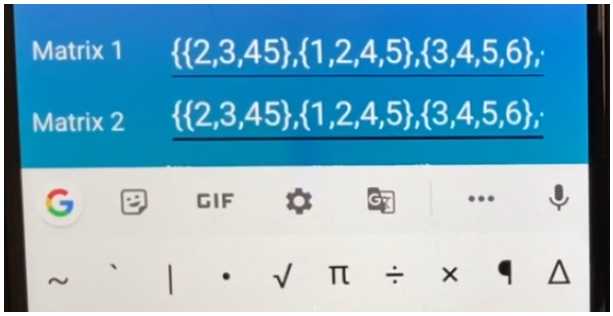

Figure 7: User has entered matrix strings into the master device

6. *Offloading the task.* Next, when ready the user presses the offload matrix calculation button, which partitions the matrix multiplication problem into submatrices and distributes them over the network to the participant devices.

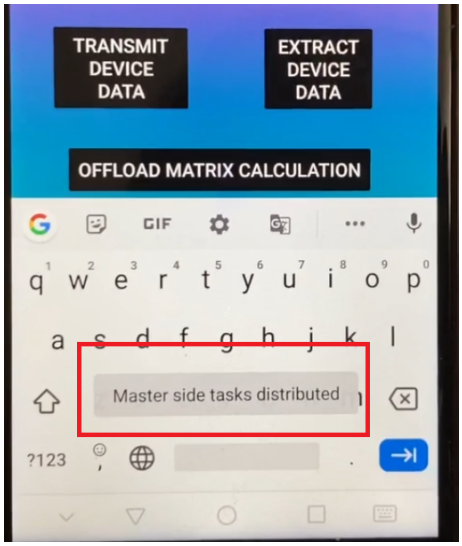

Figure 8: Master application user is notified via Toast message

7. *Performing the distributed calculations and displaying the result.* The calculations are done on the participant side, and returned to the master application where they are displayed in matrix format.

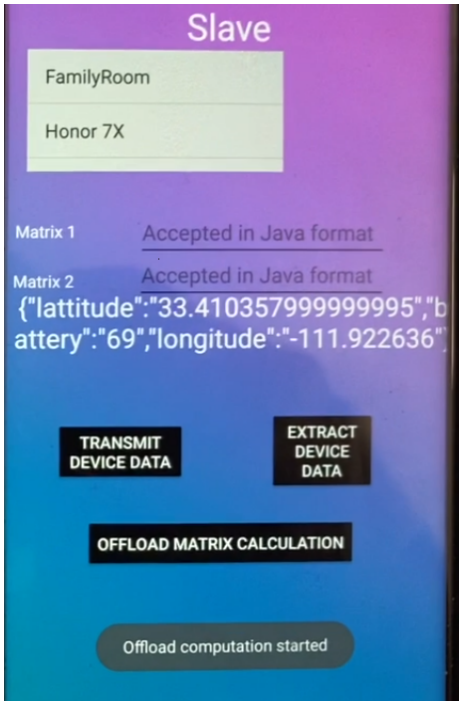

Figure 9: Participant application starting offloaded sub-matrix computation

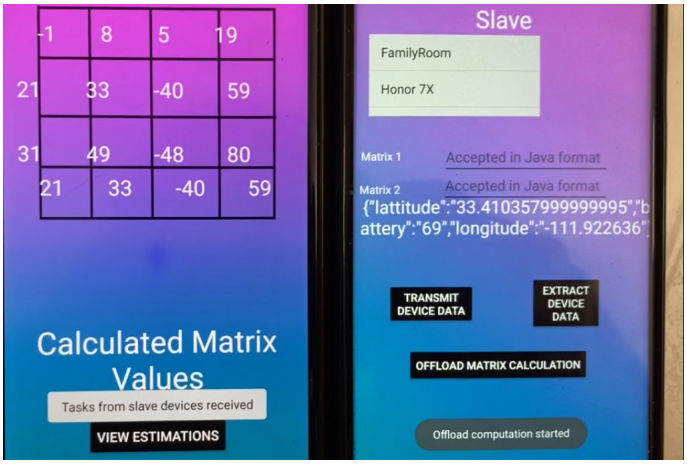

Figure 10: Finished matrix reassembled and displayed on master application

8. *Calculating estimates of power consumption.* Lastly, once completed the user may request estimate information of the power consumption of the distributed vs. non-distributed approach.

9. *Calculating estimates of execution time.* As well as the estimate of task execution time done with distributed approach, non-distributed approach, and with failure.

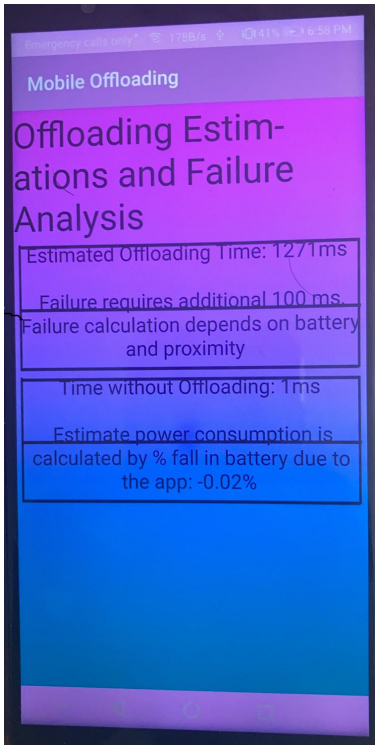

Figure 11: Estimates displayed

LIMITATIONS

Our mobile offloading application is dependent upon the availability of GPS and WiFi-direct peer-to-peer connectivity. This application's communication range is limited to approximately 100 meters at present due to limitations of WiFi signal strength and power considerations. Due to the COVID situation making acquiring additional Android mobile devices to test the application impossible, and our group having only 2 between ourselves, the testing of this application was carried out using only 2 Android mobile devices. Additionally, we could not emulate Android devices for testing this application as the framework did not support the underlying protocols our application employed, namely WiFi-Direct and GPS data. This is further complicated by emulator using the same WIFI connection for all simulated devices, limiting the ability to connect devices over WiFi.

CONCLUSION

Our team has built a mobile application that instead of taxing cloud and fog infrastructures to supplement mobile computational resources, forms a network of nearby mobile devices and allocates work among them. The allocation of work relies of metrics such as battery level and location proximity, and the declaration of one master mobile device to handle the operation (metric analysis, work allocation and distribution, and work summation with display. This is a proof of concept that nearby mobile resources can work together to form networks to solve complex computational problems without access to internet and wireless infrastructure.

Task Completion Table

Num	Task	Done
1	Master mobile application starts program that collects battery levels from mobile phones and lists it in a file	Yes
2	Application can query nearby available mobile phones through Bluetooth or WiFi to request participation	Yes
3	Application selects mobile phone among participants based on minimum battery level and location proximity	Yes
4	Application sends requests to start battery monitoring application to chosen set	Yes
5	Participant application that can receive request from master application through Bluetooth or WiFi	Yes
6	Participant application can monitor battery level and location, and send it back to master with user consent	Yes
7	Master application can tell participants to start periodic monitoring and reporting	Yes
8	Master application instantiates a matrix	Yes
9	Master application partitions matrix among participants	Yes
10	Participant application computes sub matrix multiplication	Yes
11	Participant application returns result of computation to Master	Yes
12	Master application combines sub matrices and displays finished result	Yes
13	Master application has failure recovery algorithm (if participant fails, master reassigns task to another available participant)	Yes
14	Estimate task execution time if done only on master application	Yes
15	Estimate task execution time if done with distributed approach, no failure	Yes
16	Estimate task execution time if done with distributed approach, with failure	Yes
17	Estimate power consumption of master and participants if done with non-distributed approach	Yes
18	Estimate power consumption of master and participants if done with distributed approach	Yes

REFERENCES

[1] "Android nearby api." https://developers.google.com/nearby, 2020.

[2] "Android broadcast receiver documentation." https://developer.android.com/reference/android/content/BroadcastReceiver, 2020.

[3] Sarthi, "Wifi p2p tutorial." https://www.youtube.com/playlist?list=PLFh8wpMiEi88SIJ-PnJjDxktry4lgBtN3, 2020.

[4] "Android broadcast system documentation." https://developer.android.com/guide/components/broadcasts, 2020.

[5] "Android connections api." https://developers.google.com/nearby/connections/overview, 2020.

[6] "Android location and contex api." https://developers.google.com/location-context, 2020.

[7] "Android geolocation api." https://developers.google.com/maps/documentation/geolocation/overview, 2020.

[8] "Android battery monitoring." https://developer.android.com/training/monitoring-device-state/battery-monitoring, 2020.