CSE 535: Mobile Offloading Project

Harsh Virani*, Abhilasha Mandal†, Raghav Kaul‡ and Sharadhi Jaganath§
Arizona State University
Email: *hvirani@asu.edu, †abhilasha.mandal@asu.edu, ‡rkkaul@asu.edu, §sjagana4@asu.edu

***Abstract*—This project focuses on developing a distributed computing infrastructure that can be deployed using mobile devices. A mobile offloading Android application has been created. The connection between devices is established using local Wi-Fi communication. Multiple devices within a network consent to share their battery and location information. This application employs a master-slave architecture. The master is given the task of computing the product of two matrices. This matrix multiplication problem is distributed amongst the slave devices in the vicinity of the master. Once the tasks have been distributed and the necessary computations performed, the intermediate results are sent to the master, which assimilates them and generates the final output.**

I. Introduction

Performing computations with the help of mobile devices come with some obvious advantages such as location flexibility, better time management, the convenience of use, and an increase in productivity. However, it does come at the cost of an increase in power consumption. The drain in this power directly affects the battery life of these mobile devices. One method to alleviate this problem would be to distribute the task onto multiple devices [1] within a network. The aggregation of resources combined with the distribution of necessary tasks helps to reduce the power consumption of each device significantly. Thus, a matrix multiplication task that would have initially been performed by the master device is now distributed between two devices. This helps reduce the power consumption of the master.

II. Project Setup – Permissions and Configurations

The following permissions have been enabled by the application. They are as follows:

- ACCESS_WIFI_STATE
- CHANGE_WIFI_STATE
- ACCESS_FINE_LOCATION
- ACCESS_COARSE_LOCATION
- CHANGE_NETWORK_STATE
- INTERNET
- ACCESS_NETWORK_STATE
- WRITE_EXTERNAL_STORAGE
- WRITE_INTERNAL_STORAGE

In order to accomplish the given task, the application runs Android on the given configuration.

- Compile Sdk Version 29
- Build Tools Version ”29.0.3”
- Minimum Sdk Version 21
- Target Sdk Version 29

III. Implementation

The user interface of the mobile application has been displayed in the image below. The following steps are taken to execute the required functionality.

- **Step 1** - First, all devices belonging to the network must enable their location [2] and Wi-Fi capabilities [3]. Please note that all devices must be connected [4] to the same network.

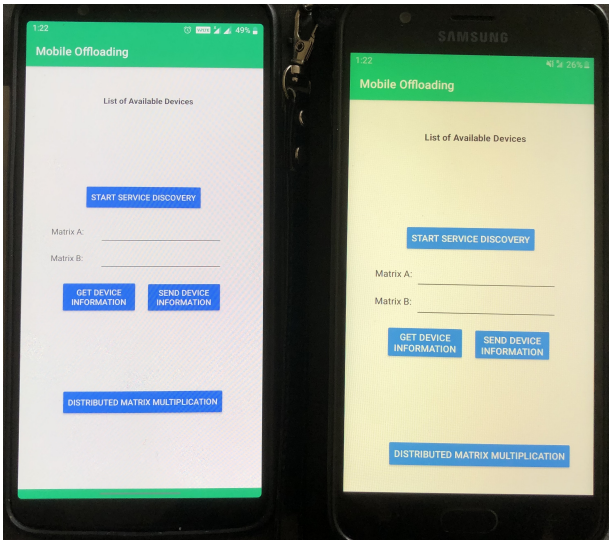

Fig. 1. Application running on two mobile devices

- **Step 2** - Next, both devices must grant access to the location sensor. This task is accomplished using the ”Get Device Information Button” on the user interface.
- **Step 3** - The service discovery application process now begins with the press of a button. Please note that the device which first sends the request [5] for service discovery becomes the master while the other mobile device becomes the slave. Now, the list of available devices is displayed on the interface, and the slave device is chosen. In our example, the name of the slave device is Samsung Galaxy J7. This action pairs both devices [6].
- **Step 4** - Next, the device information of the slave must be continuously monitored by the master. To accomplish this task, we press the ”Get Device Information” of the slave so as to retrieve its battery and location information [7]. This information is then sent to the master using the ”Send Device Information” button. As can be seen in the image, this information is displayed on the interface

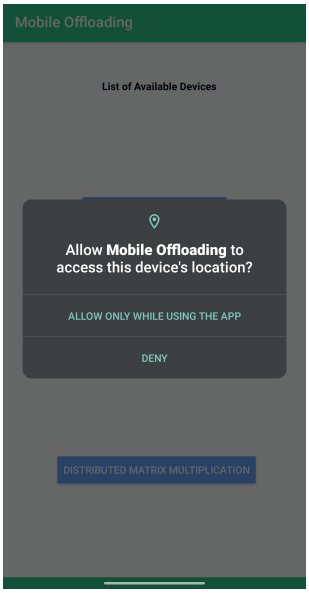

Fig. 2. Allowing device location access

Fig. 3. Pairing slave device with master

Fig. 4. Getting device information

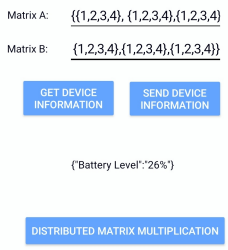

Fig. 5. Entering input matrices

of the master. The same process can be replicated by switching the master and the slave devices. However, in this case, the slave receives only the battery [8] information of the master.

- **Step 5** - We now begin the main task of distributed matrix multiplication. First, the two input matrices must be pasted onto the interface of the master. Please note that these matrices must be written per the required Java specifications. Along with this requirement, both matrices must have the correct size (number of rows and columns) and form to generate a correct result. The ”Distributed Matrix Multiplication” button is pressed offloading this task to the slave device.
- **Step 6** - Once the slave device performs the necessary calculations, it returns the result back to the master. The resultant matrix is displayed on a fresh page by the master device.
- **Step 7** – Finally, the standalone and distributed execution times along with the distributed power consumption is also displayed on the new page. We have noticed for a problem as simple as matrix multiplication, the precision of the android device is not small enough to reflect changes in the standalone and distributed power consumption.

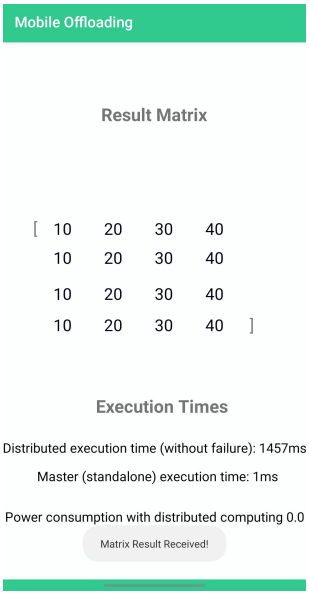

Fig. 6. Obtaining the result

IV. Completion of tasks

No	Task Name	Assignee
1	Writing a mobile application that collects the battery levels of a mobile phone and lists it in an app.	Raghav K, Harsh V
2	Developing a service discovery application that sends queries to mobile phones in close proximity through Wifi to request participation.	Raghav K, Harsh V
3	Developing a dispatch application that chooses one particular device amongst the host of mobile devices that accept the request for participation.	Raghav K, Harsh V
4	Developing an application that requests for the battery information of slave devices from the master device.	Raghav K, Harsh V
5	Developing a slave device application which can accept a request for battery information from the master device.	Raghav K, Harsh V
6	Extending this application by creating a feature that allows the slave device to send both battery and location information back to the master device once such a request by the master has been accepted.	Raghav K, Harsh V
7	Extending the above feature to begin a periodic monitoring of the requested states (location and battery information).	Raghav K, Harsh V
8	Replicating this infrastructure to offload a matrix multiplication problem to the slave device.	Abhilasha M, Sharadhi J
9	Estimation of execution time of matrix multiplication if done only by the master.	Abhilasha M, Sharadhi J
10	Estimation of execution time of matrix multiplication if distributed between the master and the slave without any failure.	Abhilasha M, Sharadhi J
11	Estimation of execution time of the matrix multiplication of distributed between the master and the slave with failure.	Abhilasha M, Sharadhi J
12	Estimation of the power consumption of the master and slave nodes without computational distribution.	Abhilasha M, Sharadhi J
13	Estimation of the power consumption of the master and slave nodes with computational distribution.	Abhilasha M, Sharadhi J
14	Developing the user interface of the entire application.	Abhilasha M, Sharadhi J

V. Limitations

Perhaps the first and most limiting constraint of our application is the necessity of all mobile devices in the network to have both GPS and Wifi-direct features. Without these features, none of the applications mentioned in the paper can be performed on them or by them. Next, keeping in mind the power drainage capabilities of these applications, one constraint has been placed on which slave devices can be chosen to offload the task to. This constraint is that the battery of the slave device should be more than twenty percent. Without this criteria met, the slave node will not be chosen despite its presence in the network. Due to the unavailability of resources, our application was tested only using two mobile devices - a master and a slave. The natural extension of our work would be to test this capability on multiple slave nodes.

Another limitation of our project is that the matrices cannot have a size greater than 4x4. Our application could have been further burnished by the addition of a feature that could test whether the two input matrices were indeed in accordance with Java specifications. For further enhancement of user experience the user interface can be improved by providing an option to first input the size of the two matrices and then allowing the user to input these numbers directly into a box shaped like a matrix.

VI. Conclusion

The central idea of this project is to take a limitation of using mobile devices (its power usage) and convert it into a strength. By employing multiple devices to perform a particular task, not only is there a great reduction in the power usage of

individual devices within the network, but also the ability to pool resources from all these devices to perform the required task. As these tasks can be performed using only Wifi-direct and GPS capabilities, we do not require high-speed internet connectivity or cloud infrastructure to accomplish our goal. We are also able to numerically quantify the reduction in power consumption and execution time by offloading the task.

VII. Acknowledgement

We would like to take this opportunity to offer our sincere gratitude to Professor Ayan Banerjee for allowing us to work on this project. Through the process of implementing all the requirements of this application, several concepts have become clear to us, greatly steepening our learning curve within a short amount of time.

References

[1] *"Android nearby api." https://developers.google.com/nearby, 2020.*
[2] *"Android broadcast system documentation." https: //developer.android.com/guide/components/ broadcasts, 2020.*
[3] *https://www.youtube.com/channel/UCPM25ORxxFAlTGA6u2Vj-5w.*
[4] *"Android broadcast receiver documentation." https://developer.android.com/reference/ android/content/BroadcastReceiver, 2020.*
[5] *Android geolocation api." https:// developers.google.com/maps/documentation/ geolocation/overview, 2020.*
[6] *"Android connections api." https:// developers.google.com/nearby/connections/ overview, 2020.*
[7] *"Android location and contex api." https:// developers.google.com/location-context, 2020.*
[8] *"Android battery monitoring." https:// developer.android.com/training/monitoringdevice-state/battery-monitoring, 2020.*