Real-Time Mobile Offloading for Matrix Multiplication

CSE 535 Mobile Computing (Spring 2020)

Kusumakumari Vanteru
kvanteru@asu.edu
1217031218

Tejaswi Paruchuri
tparuchu@asu.edu
1213268054

Thanuja Kallamadi
tkallama@asu.edu
1217132202

Snehitha Rednam
srednam@asu.edu
1217164507

***Abstract* - With the advent of high connectivity among devices, task distribution among various devices is gaining popularity nowadays. Distributing tasks among other devices reduces the load on a single device and gives us higher performance with lesser battery utilization. Increased availability of devices would increase the efficiency in dealing with bigger loads. In this paper we demonstrate mobile offloading using a matrix multiplication problem. The mobile offloading problem follows a master slave approach to distribute tasks and communicate effectively. The dispatcher application for redistributing the workload at the master works by considering the battery level and current location of the devices. Also, we effectively devise failure recovery algorithms by reassigning to other resources.**

Keywords - Mobile Offloading, Bluetooth, Master-Slave, Firebase Cloud Messaging, Android, mobile application

I. INTRODUCTION

In the past few years, there have been many developments in the hardware and software for smartphones and similarly a variety of applications are being developed and used like Google Translate, photo, audio and video processing apps, games with high-end graphics, etc. All these require large amounts of computational power, battery and RAM usage. These kinds of tasks would be done much faster if we adopt the concept of Mobile Offloading. The basic idea is to distribute the tasks that require lots of resources, by a Master device to all the other devices connected to it in a local network. This would be like a local fog server connected via either Bluetooth or Wi-Fi, in which one node is the Master and the rest are Slaves. The main advantage of this is that it reduces the load on cellular networks and hence, would result in a better network bandwidth.

However, data security is the major concern always. According to Z. Zhang et. al [1], we choose to avoid the part of data offloading that deals with confidential data that cannot be protected in the other devices. Also, we have different kinds of offloading strategies [1] like Virtual machine migration, entire application migration, application partitioning meaning the tasks are separated at runtime, etc. In this project, we install the application in all the mobile devices and the first one to search for other devices to connect becomes the master node. Some of the challenges in this field include fault tolerance, mobility, dealing with big data streams, security of private data, location, etc.

This project discusses in detail about designing and developing such a mobile application, which allows the Master node to allocate tasks, collect the results and display the statistics like battery level drop, time taken, etc.

II. PROJECT SETUP

In order to develop this mobile application in the Android Studio IDE in a computer, we need the below prerequisites:

Android: Version: Minimum version 26

*Local Server***:** Apache Xampp v3.2.4

Operating System: Windows 10

RAM: 16 GB

Hard disk: 1 TB

*Firebase Cloud Messaging Server***:** This server helps the master to send push notifications to the devices having the application installed. The Firebase Cloud Messaging service is a cloud-based service provided by Google which provides Mobile backend as a Service (MBaaS).

The below figure represents a brief architecture of our system, where we have the Master device connected to all the other Slave devices using Bluetooth, and Master communicates with others by sending instructions through messages over the Bluetooth network. The Firebase Cloud server would accept the package name of the app and give the authorization token, that would be in the local server startup page. As soon as the Master requests the server to send notification messages, the FMC server sends notifications accordingly, to all the slave nodes.

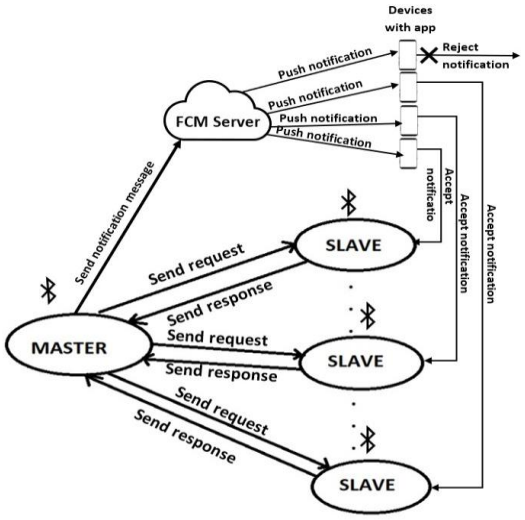

Fig. 1: Architecture of the system

III. IMPLEMENTATION

Once the application was developed using the code references [2], [3], [4], [5], we had to test it. For implementing our whole mobile offloading application, we first talk about the initial installations that need to be done while hosting the app:

- The APK of the app needs to be installed in each of the mobile devices
- The Apache tomcat server should be started, in order to call the FMC server
- Location permission should be enabled in all the mobile devices

Various components that are present on the screen of the application are as below:

1) Battery Level Text View:
   This text view shows the battery level of the device.
2) Connection Status Text View:
   This text view shows the information regarding the Connection status of the device.
   Some of the Connection statuses available are Connecting, Connection failed, Connected, Battery Low can't connect, Disconnected.

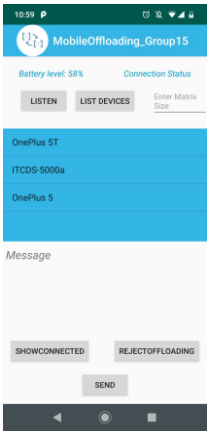

Fig. 2: User Interface of the application

3) Listen Button:
   This button is used by the slave if it is willing to connect to the Master device.
4) List Devices Button:
   This button helps the master to see the nearby available Bluetooth enabled devices.
5) Enter Matrix Size Edit Text:
   This Edit text is used by the master to enter the size of the matrix for which random matrices are to be generated. It remains disabled if the device is a slave.
6) List view of devices:
   Displays all the nearby available Bluetooth devices in the list format.
7) Message Text view:
   This text view shows the result of the output matrix along with the statistics related to battery level drops and computation times taken. It also displays the devices connected and battery levels. At the slave end the information related to the rows received can be seen.
8) Show connected button:
   This button is used to see the devices which are connected to a particular device.
9) Reject Offloading button:
   This button is used by the slave device if it is not willing to do the computation at its end. It remains disabled at the master end.
10) Accept Offloading button:
    This button is used by the slave device if it has clicked reject offloading and is again willing to accept the offloading tasks. This button remains disabled at the master end.

11) Send button:
This button is used by the master device to send the matrices to its slaves and the button remains disabled at the slave end.

Now, let's discuss the different use-cases possible and how the mobile app would work in those scenarios:

A. Use case 1 - Master sends matrix size and Slaves send the result

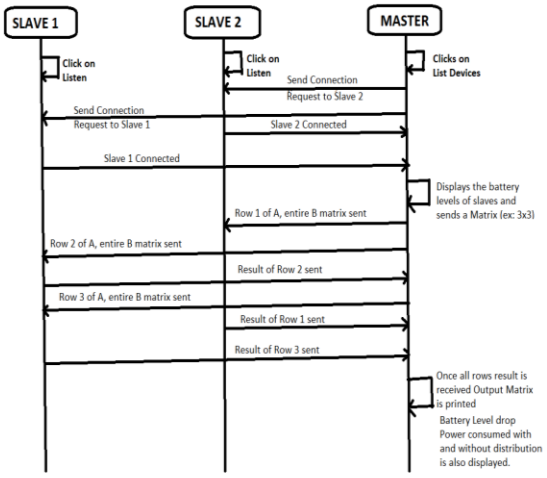

Fig. 3: Sequence diagram of Master offloading tasks to the slave

First a device is selected as a master device and the list devices button is clicked. On click of the list devices button a pop-up notification is sent to all the devices which have installed the Application asking if they could help the master in matrix multiplication. As soon as the application is open in case if Bluetooth is disabled it will be enabled.

If the device is willing to help it clicks on the notification and clicks on the listen button. Now this device starts acting as a slave and master can connect to it with Bluetooth by clicking on the device name in the list of devices.

On successful connection the master receives the battery level and location information of the slaves.

The master can then send the matrices of size nxn to the connected slaves to do matrix multiplication. A validation check is done at the master end to see that the matrix size is not null and is between 2 and 12.

Whenever the size entered by the master is with the limits two random matrices are generated at the master end of size nxn. Each row of the first matrix along with the entire second matrix is sent to its slave devices along with row number. The slaves calculate the result and send back the result to the master along with row number. Once the master receives the result from a particular slave it checks if there are any other rows which need to be calculated. If there are any such rows, they are again sent to the slaves which have returned their previous results.

Once the results of all the rows are calculated the output i.e. multiplied matrix is displayed on the master device screen.

The battery level drops at all the slaves and master due to the mobile offloading, is displayed at the master. The battery level drop when the computation is done only at the master end without offloading is also displayed. The time taken (in nanoseconds) with and without offloading results are also displayed.

B. Use case 2 - One of the Slave rejected offloading

When the slave is not willing to do the offloading tasks at any point of time while listening it can click on the Reject offloading button. Once the Reject offloading button is clicked it turns into the Accept offloading button. When the slave again wishes to do the offloading tasks, it can click on the accept offloading button.

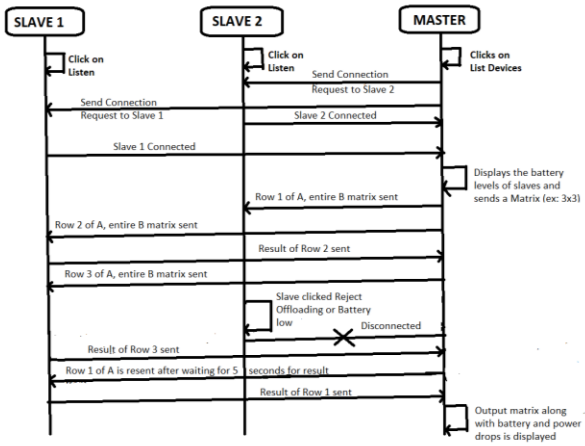

Fig. 4: Sequence diagram of Slave rejecting offloading tasks sent by the Master

C. Use case 3 - Battery Low at the Slave

The battery threshold value is set to 40%. Only when the slave's battery is above 40% will they be able to calculate the matrix multiplication tasks. If the battery drops in the middle when the task is running then the failure recovery

algorithm comes into picture and the low battery slave gets disconnected. The master waits for the result from the low battery device for 5 seconds and resends the tasks sent to the low battery device to other connected devices.
If the battery is low initially at slave the connection is not established between the master and slave though the master tries to get connected.

D. Use case 4 - Battery Low at the Master

The battery threshold value is set to 40%. The master will be able to send matrices for calculation only when the battery of the master is greater than 40%. If the battery is below the threshold value a message is displayed at the master end that the battery is low.

E. Use case 5 - Slave is not in the proximity of Master

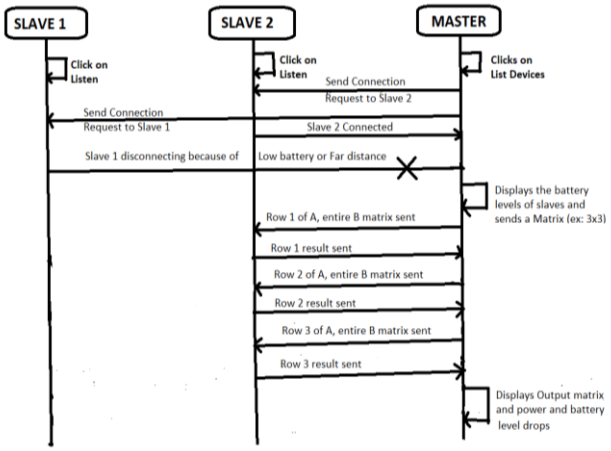

Fig. 5: Sequence diagram of Slave rejecting offloading tasks sent by the Master

A connection is established between the master and slave only when the slave is within the proximity of the master. In this application it is set to 0.1 miles. Only when the slave is within this radius it gets connected to the master. If not, a message is displayed at the master end that the slave device is far and cannot be connected.

IV. RESULTS

Below are some of the screenshots of the when the mobile application was tested under various conditions:

Fig. 6: Screenshot of pop up notification requesting the slave for the help

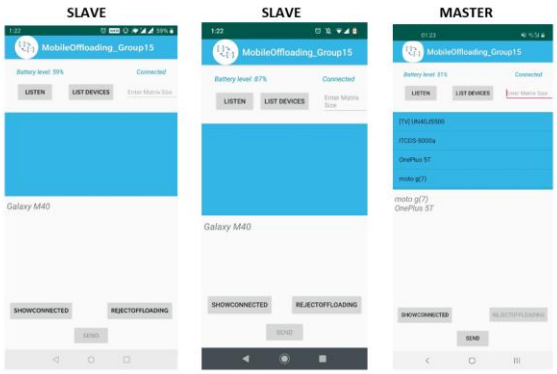

Fig. 7: Screenshots of Master connected to two Slave devices

A. Use case 1 - Master sends matrix size and Slaves send the result

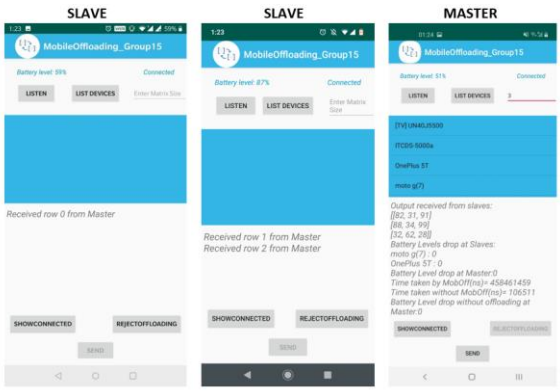

Fig. 8: Screenshots of Master receiving the output result matrix from its connected two Slave devices

The above images display the screenshots of the master and slaves when a 3x3 matrix is sent from the master to its connected slaves.

B. Use case 2 - One of the Slave rejected offloading

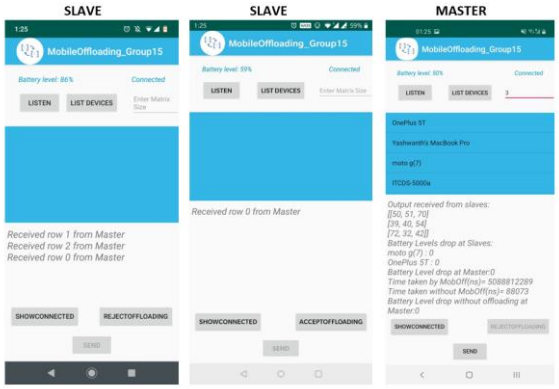

Fig. 9: Screenshots of Master receiving the output result matrix when one Slave device clicks rejects offloading

The images above show the screenshots of the master and slave when the master sends a 3x3 matrix but one of its devices clicked the reject offloading button.

C. Use case 3 - Battery Low at the Slave

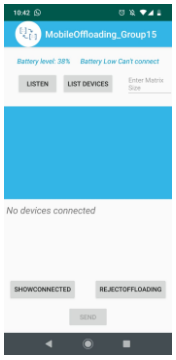

Fig. 10: Screenshot of Master showing its battery is low

The image above shows the screenshot of the slave when its battery level is low.

D. Use case 4 - Battery Low at the Master

The below image shows the screenshot of the master when its battery is low.

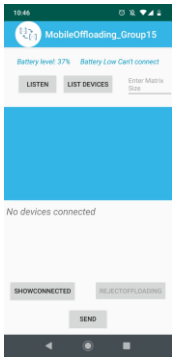

Fig. 11: Screenshot of Slave showing its battery is low

E. Use case 5 - Slave is not in the proximity of Master

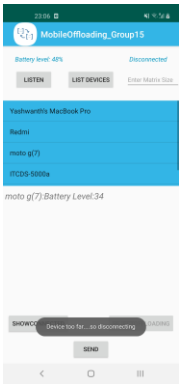

Fig. 12: Screenshot of Master displaying that the slave device is far to connect

The above image shows the screenshot of the master when the slave which master is trying to get connected is far and is not in the proximity.

V. DISTRIBUTION OF TASKS

Task	Done by	Status
Master Mobile application that collects battery levels from phones and lists them	K.V, T. P, T. K, S. R	Done
Service Discovery application that sends queries to nearby available phones through Bluetooth or Wi-Fi, to request participation	K.V, T. P, T. K, S. R	Done
Dispatcher application that choses	K.V, T. P,	Done

a mobile phone that accepted the request based on some matching criteria	T. K, S. R	
Based on chosen set, send requests to start battery monitoring application on the slave side	K.V, T. P, T. K, S. R	Done
Slave application that can receive a request from a master through Bluetooth or Wi-Fi	K.V, T. P, T. K, S. R	Done
Monitor battery level and current location and send it back to the master if the user decides to consent	K.V, T. P, T. K, S. R	Done
Application to run a code snippet in Slave to start periodic monitoring	K.V, T. P, T. K, S. R	Done
Solve the problem of distributed Matrix Multiplication	K.V, T. P, T. K, S. R	Done
Failure recovery algorithm in the Master, where if a slave fails the master can reassign immediately to another available slave node	K.V, T. P, T. K, S. R	Done
Estimate execution time of the Matrix Multiplication if done only on the Master	K.V, T. P, T. K, S. R	Done
Estimate execution time If done using the distributed approach with no failure	K.V, T. P, T. K, S. R	Done
Estimate execution time if done using the distributed approach with failure	K.V, T. P, T. K, S. R	Done
Estimate power consumption of Master and Slave nodes without distributed computation	K.V, T. P, T. K, S. R	Done
Estimate power consumption of Master and Slave nodes with distributed computation	K.V, T. P, T. K, S. R	Done

Table 1: Distribution of tasks

The abbreviations indicate the authors as follows:
K. V: Kusumakumari Vanteru, T. P: Tejaswi Paruchuri, T. K: Thanuja Kallamadi, S. R: Snehitha Rednam

VI. LIMITATIONS

Our application performs efficiently with a matrix size of 2 to 12 which can be extended in future. Our application is best recommended for devices with an android version greater than 26, but works with other versions as well with few limitations. Our application works only on the Bluetooth enabled devices with GPS feature availability. In some cases, GPS might not be accurate enough to give latitude and longitude because of which the master may not be able to connect to slaves.

VII. CONCLUSION AND FUTURE WORK

The mobile offloading concept has effectively helped us in dealing with the matrix multiplication problem by distributing the task among the slave devices and in future we can extend to bigger dimensions. We can expand the utilization of the mobile offloading concept by applying in various real-world applications of gaming, mobile data offloading etc. by scaling and optimizing networks. Also, with the advent of cloud, mobile offloading will get a newer dimension in dealing with resource utilization. In the future with much more connectivity and higher speeds mobile offloading would definitely be the norm of all the tasks performed in a device.

VIII. ACKNOWLEDGEMENT

We would like to thank our professor Dr. Ayan Banerjee for encouraging us to take up the project on Mobile Offloading and for continuously providing inputs and resolving our queries.

REFERENCES

[1]: Z. Zhang, S. Li, "A Survey of Computational Offloading in Mobile Cloud Computing", IEEE International Conference on Mobile Cloud Computing, Services, and Engineering, 2016
[2]: Super Coders, "https://www.youtube.com/watch?v=1lT0ZliubU0", November, 2019
[3]: Geodata Source, "https://www.geodatasource.com/developers/java", 2019
[4]: Tutorials Point Developers, "https://www.tutorialspoint.com/android/android_location _based_services.htm", 2019
[5]: Sarthi Technology, "https://www.youtube.com/watch?v=NR1sDXMzyww&list=PLFh8wpMiEi8_I3ujcYY3-OaaYyLudI_qi", April 2017