# Mobile OffLoading

Pawan Wagh
*Arizona State University*
1219432396
pwagh2@asu.edu

Varun Kumar Reddy G,
*Arizona State University*
1222362973
vgunnred@asu.edu

Rajashekar Reddy A
*Arizona State University*
1217211645
raluka@asu.edu

Venkata Pavana Kaushik N
*Arizona State University*
1223627444
vnandula@asu.edu

***Abstract*—The main of the Mobile OffLoading project is to design and develop a distributed android application which does matrix multiplication. The mobiles connect each other using Wifi or Bluetooth to do the communication between devices. In our application we have master slave architecture where one of the devices will act as master which distributes to the work to slaves to do the matrix calculation work by forming network like fog cloud. In this we project we did experiment's of computing a 100x100, 500x500 and 1000x1000 matrix multiplications.**

***Index Terms*—Mobile Computing, Fog cloud, Distributed Computing, Android, Bluetooth, Wifi**

## I. Introduction

As in today's world each every individual mostly use a mobile device for many purposes. The computing power of mobile device is limited and many times a user might not be able to do all the computations that he wants to do. In areas with low internet, the most possible way to do complex computing work is through using distributed computing over mobile devices by connecting via simple Wifi or Bluetooth. Mobile offloading is one example of how can we do distributed computing using mobile devices.

Using mobile offloading techniques, we can reduce the load on cellular networks. This technique can be used to design distributed applications. Architectures like Master-Slave(also known as primary-secondary) can be used with offloading concept to develop distributed applications. In case of Master-Slave, master can be any device which needs to get some task performed.
We have used the same concept to build a application which can perform matrix multiplication with n*n*n* upper bound. Large sized matrices can be divided and multiple devices can work on it at the same time. Results obtained from each device can be fused together to get the final product.
Master device connects with slaves which are in vicinity of master using WiFi and Bluetooth. There has to be some threshold for battery level. Master divides the work between slaves and slaves perform the operation on received data. Output of the operation is transferred back to master. Work piece allocated to slave device goes back to master device if any slave device is disconnected or its power level falls under certain limit. This work is again send to connected nodes to perform computation.

## II. Project Setup and Permissions

The following setup consists of three components.

- Android devices ( 3-5 devices) where one device will act as master and the remaining devices will act as slave which communicates over Bluetooth or Wifi.
- Android Version: Android 9.0 (Minimum API level 23, Target API Level 29)
- Android Studio 3.0 or higher

### *A. Permissions Required on the mobile phone are:*

- BLUETOOTH BLUETOOTH_ADMIN
- ACCESS_COARSE_LOCATION
- ACCESS_WIFI_STATE
- ACCESS_FINE_LOCATION
- CHANGE_WIFI_STATE

API: Google Nearby Connections API [1] Google Play, Services Location [2]

## III. System Architecture Diagrams

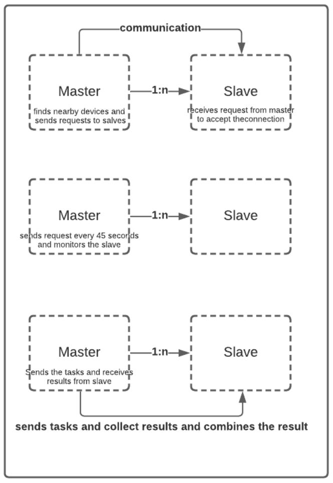

communication
Master
finds nearby devices and sends requests to salves
1:n
Slave
receives request from master to accept theconnection
Master
sends request every 45 seconds and monitors the slave
1:n
Slave
Master
Sends the tasks and receives results from slave
1:n
Slave
sends tasks and collect results and combines the result

Fig. 1: Mater Slave architecture

The figure. 1 shows the architecture design of the Mobile Offloading application.

## IV. Implementation

### A. Establishing connections between Master and Slave

On entering to the app, the home screen will show 2 options. We can select either a master or slave.

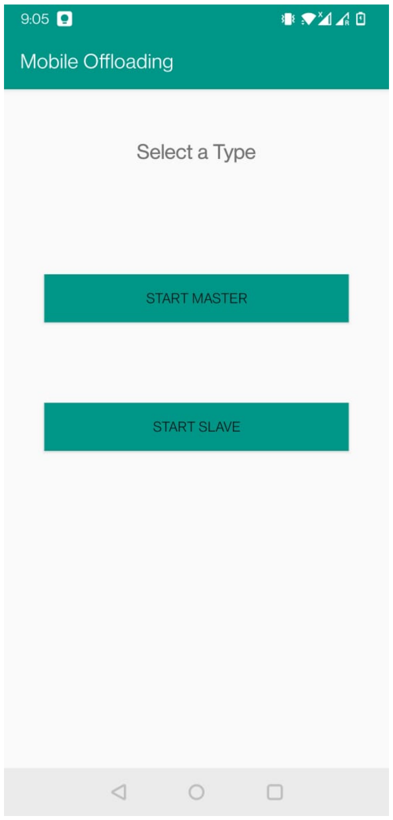


Fig. 2: Home App Screen

The below image shows the screen of master app where on clicking find slaves will try to discover the mobiles in nearby location which are with in the range of 100m.

Now the slaves receives the request from the master and on accepting it we show mobileId, battery and status whether they are connected or not.

### B. Tracking battery Status

Master device tracks battery status of slave nodes. Master sends requests to slave devices at intervals of five seconds. Response from slave device includes identifier, remaining power level, location coordinates.This information is useful in deciding whether to disconnect a slave or keep sending work to it. thresholds for battery and proximity to master are thirty percent and hundred meters respectively. Results obtained from regular fetch operations are displayed in the below figure.

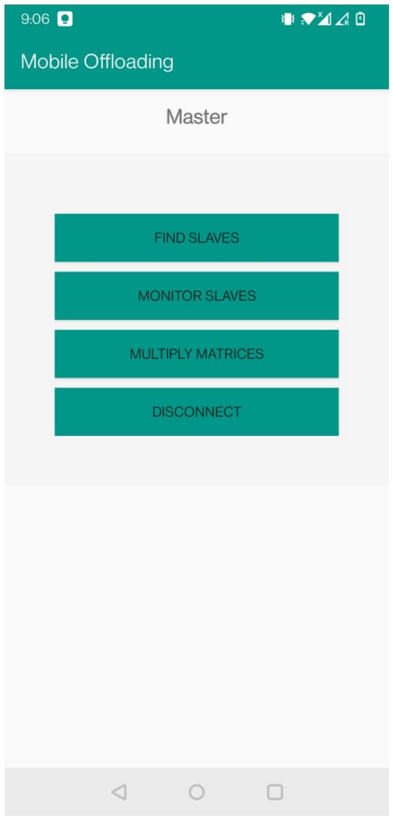


Fig. 3: Mater Screen

### C. Multiplication of matrices in Distributed Way

For demonstration, Master creates two matrix when multiplication operation is requested on application. generated matrices has 1000 rows and 1000 columns. In case when slave devices are not connected to master, this operation is cancelled and a popup message is displayed to the user. When connected slave meets battery and distance thresholds, entry for this slave is created as available slave.

Master starts task to track slave devices. Errors and delayed results have to be handled by this task. For dividing matrices, partitions have to be created. These are created by using indices. Indices are stored to identify unique partitions. Once partitions are created and we have connected to slave devices, this work can be distributed among them. Partition list can be considered as task list for the slaves and this tasks can be assigned among all the available slaves. Inverse mapping between slave identifier and partition is maintained too. In case of task failures, this mapping can help in failed partition and it can be assigned to some other slave. Slave receives a task to be performed in JavaScript object notation format. Parsing of these objects is done to get the partitions of the matrices. When multiplication is done, matrices are converted back to JSON and slave starts uploading

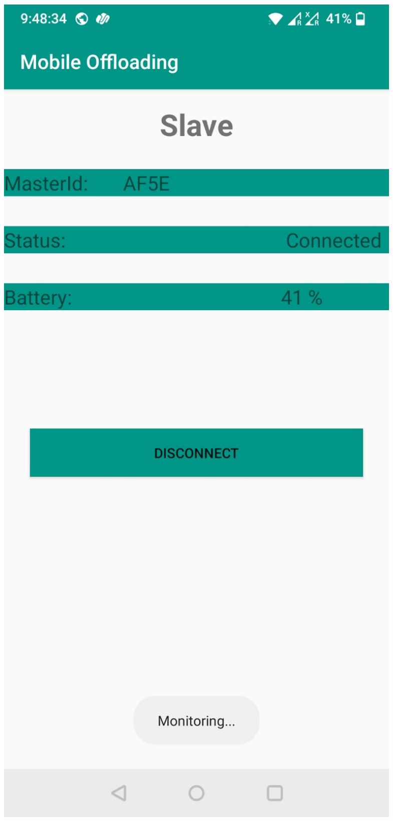


Fig. 4: Slave Screen

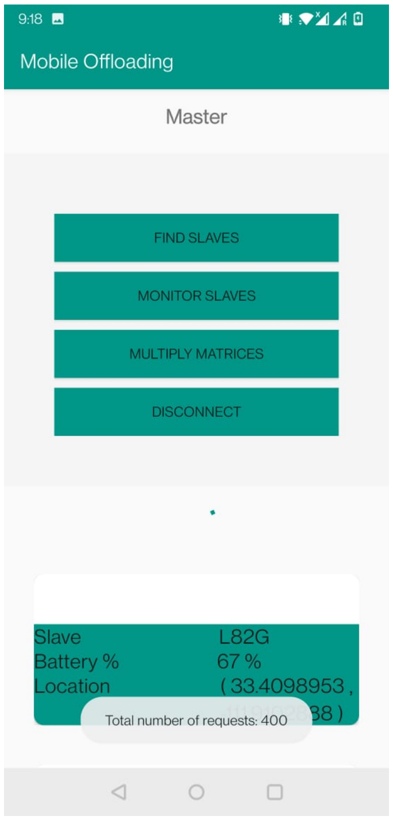


Fig. 5: Battery Status

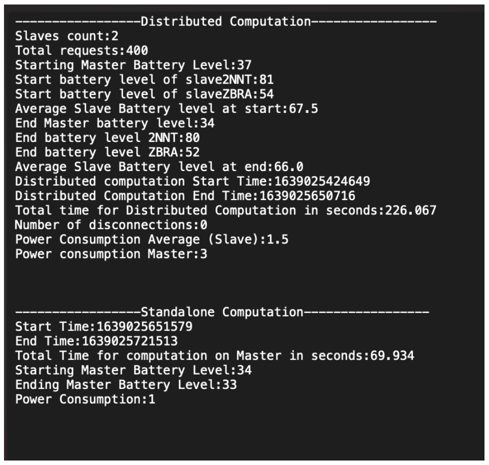
these to master device. Master keeps assigning tasks to available slaves and combines results. Application keeps updating users with operation progress based on results received and work remaining. Popup message is displayed to the user after completion of multiplication.

### D. *Handling Failures in Distributed Computing*

Main challenges in the distributed application involve failures due to network parameters such as bandwidth, packet loss, connectivity. Distributed systems should handle failures due to network partitions and recover from them. Failure of single slave shouldn't affect the entire system. Our application implements tracking mechanism to keep checking status of connected slaves. Last connection time of each slave is maintained and if connection time goes above certain threshold, status of slave is changed to stale. Since slave is not reachable, master tries to recover from this failure and assigns that slave's partition to other device which is available. If any device is not available immediately, then this work is added to the task list maintained by the master. Stale slave is deleted from the available devices to prevent assignment of further tasks which could result into failure. Statistics of error resolution timing is also maintained which can help in further analysis of error handling.

### E. *Execution of time for the tasks*

For our experiment we are using 3 mobile devices where 1 device is master and 2 as slaves to do the matrix multiplication in various scenarios.



```
-----------------Distributed Computation-----------------
Slaves count:2
Total requests:400
Starting Master Battery Level:37
Start battery level of slave2NNT:81
Start battery level of slaveZBRA:54
Average Slave Battery level at start:67.5
End Master battery level:34
End battery level 2NNT:80
End battery level ZBRA:52
Average Slave Battery level at end:66.0
Distributed computation Start Time:1639025424649
Distributed Computation End Time:1639025650716
Total time for Distributed Computation in seconds:226.067
Number of disconnections:0
Power Consumption Average (Slave):1.5
Power consumption Master:3



-----------------Standalone Computation-----------------
Start Time:1639025651579
End Time:1639025721513
Total Time for computation on Master in seconds:69.934
Starting Master Battery Level:34
Ending Master Battery Level:33
Power Consumption:1
```

Fig. 6: Benchmark Results

| Matrix Size | 1000x1000 |
|---|---|
| Number of Slaves | 2 |
| Time taken for computation only on Master | 69.934 sec |
| Time taken for distributed without failure | 226.07sec |
| Time taken for distributed with failure | 411 sec |

TABLE I: Task Time Execution Estimation

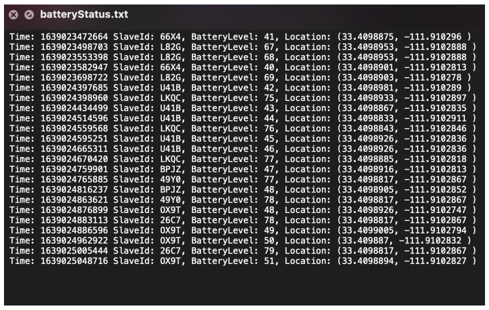
The following are the scenarios we have considered, and the following are the times we collected during our experiment.

- Matrix Computation only master.
- Distributed matrix multiplication with 2 slaves without failure
- Distributed matrix multiplication with 2 slaves with failure In this scenario during matrix multiplication, we have disconnected 1 slave with the master during the computation. We are also storing the disconnected information of the slave in batterystatus.txt file. From this information we are calculating the start time and end time which we are using it to calculate the time taken for the task to complete using the formula.
  Current Time = System.currentTimeMillis(). Execution Time = End time - Start time

In this we have used 2 slaves, on disconnection of a slave only one slave does the matrix computation which took around 411 seconds.

### F. *Power Estimation by Slaves*

We have performed various matrix multiplications of sizes 100x100, 500x500 and 1000x1000. In this report we will be reporting the time taken for 1000x1000 matrix and as well as the power consumption. We are recording 2 kinds of stats into the files, one is battery status and other is benchmarks as shown in the figure. we this stats from the files to calculate the estimation of power used in milliAmpere-hour (mAh) for doing the computation of matrix multiplication.



```
batteryStatus.txt
Time: 1639023472664 SlaveId: 66X4, BatteryLevel: 41, Location: (33.4098875, -111.910296 )
Time: 1639023498703 SlaveId: L82G, BatteryLevel: 67, Location: (33.4098953, -111.9102888 )
Time: 1639023553398 SlaveId: L82G, BatteryLevel: 68, Location: (33.4098953, -111.9102888 )
Time: 1639023582947 SlaveId: 66X4, BatteryLevel: 40, Location: (33.4098901, -111.9102813 )
Time: 1639023698722 SlaveId: L82G, BatteryLevel: 69, Location: (33.4098903, -111.910278 )
Time: 1639024397685 SlaveId: U41B, BatteryLevel: 42, Location: (33.4098981, -111.910289 )
Time: 1639024398960 SlaveId: LKQC, BatteryLevel: 75, Location: (33.4098933, -111.9102897 )
Time: 1639024434499 SlaveId: U41B, BatteryLevel: 43, Location: (33.4098867, -111.9102835 )
Time: 1639024514596 SlaveId: U41B, BatteryLevel: 44, Location: (33.4098833, -111.9102911 )
Time: 1639024559568 SlaveId: LKQC, BatteryLevel: 76, Location: (33.4098843, -111.9102846 )
Time: 1639024595251 SlaveId: U41B, BatteryLevel: 45, Location: (33.4098926, -111.9102836 )
Time: 1639024665311 SlaveId: U41B, BatteryLevel: 46, Location: (33.4098926, -111.9102836 )
Time: 1639024670420 SlaveId: LKQC, BatteryLevel: 77, Location: (33.4098885, -111.9102818 )
Time: 1639024759901 SlaveId: BPJZ, BatteryLevel: 47, Location: (33.4098916, -111.9102813 )
Time: 1639024765885 SlaveId: 49Y0, BatteryLevel: 77, Location: (33.4098817, -111.9102867 )
Time: 1639024816237 SlaveId: BPJZ, BatteryLevel: 48, Location: (33.4098905, -111.9102852 )
Time: 1639024863621 SlaveId: 49Y0, BatteryLevel: 78, Location: (33.4098817, -111.9102867 )
Time: 1639024876899 SlaveId: OX9T, BatteryLevel: 48, Location: (33.4098926, -111.9102747 )
Time: 1639024883113 SlaveId: 26C7, BatteryLevel: 78, Location: (33.4098817, -111.9102867 )
Time: 1639024886596 SlaveId: OX9T, BatteryLevel: 49, Location: (33.4099005, -111.9102794 )
Time: 1639024962922 SlaveId: OX9T, BatteryLevel: 50, Location: (33.409887, -111.9102832 )
Time: 1639025005444 SlaveId: 26C7, BatteryLevel: 79, Location: (33.4098817, -111.9102867 )
Time: 1639025048716 SlaveId: OX9T, BatteryLevel: 51, Location: (33.4098894, -111.9102827 )
```

Fig. 7: Benchmark Results

| Mode of Task | Master | Slave |
|---|---|---|
| Entire Computation on Master | 33 mAH | 0 |
| Computation Distributed between a Master and 2 Slaves | 0 | 26.63 mAH |

TABLE II: Power Consumption Estimation with out failure

| Mode of Task | Master | Slave |
|---|---|---|
| Entire Computation on Master | 33 mAH | 0 |
| Computation Distributed between a Master and 2 Slaves | 0 | 71.68 mAH |

TABLE III: Power Consumption Estimation with failure

## V. Division of Work and Team Members' Contributions

Table I shows the details about the division of task among the team. Individual contribution details of the team members are mentioned below:

## VI. Limitations

In the developed application, the master distributes the tasks of matrix multiplication to nearby devices to do the computation faster. There might some overhead and limitations like

- Network Congestion and Network failures
- Task Distribution Overhead
- Message Transmission Time

we observed that using this approach for small computation might cause unnecessary overhead in the distribution of tasks. We also noticed that increasing the number of clients/slaves might cause slave failures. The failures are because of network congestion.

## VII. Conclusion

This project involves in developing an application that executes the Matrix Multiplication Tasks using Master-Slave Architecture. In this experiment we explored various Mobile offloading techniques, understanding their significance and importance by learning the demand for computation. Then, we analysed the idea of Master-Slave architecture to perform different task using the computation power of devices present in certain proximity. This Model of Master-Slave works in such a style that tasks were distributed among slave devices which are near by in-order to increase the computation speed. By using the distributed computing mechanism especially for the matrix multiplication tasks, which involves in the breakdown of the problem among different slave devices to improve the computation power using offloading framework.

| Task No. | Task Name | Assigned to |
|---|---|---|
| 1 | Master Application - Collect battery level from mobile phones and list it in file | Pawan Wagh |
| 2 | Slave discovery - send query through WIFI or Bluetooth to request participation | Pawan Wagh |
| 3 | Slave Checking the Threshold Battery status and Location proximity to accept/reject the device | Pawan Wagh |
| 4 | Raise a Request to start Battery Monitoring on slave device | Pawan Wagh |
| 5 | Slave Application - To receive a request from Master via Bluetooth / WiFi | Rajashekar Reddy Aluka |
| 6 | Application to send battery Level and location from Slave to Master | Rajashekar Reddy Aluka |
| 7 | Application to start periodic monitoring in the slave end. | Rajashekar Reddy Aluka |
| 8a | Solving problem distributed Matrix Multiplication by instantiating | Rajashekar Reddy Aluka |
| 8b | Sending Matrix form Master to slave | Venkata Pavana Kaushik |
| 8c | Computation of Matrix at Slave End | Venkata Pavana Kaushik |
| 8d | Sending Computed Data to Master from Slave | Venkata Pavana Kaushik |
| 8e | Combining Data and Projecting Results on Master end. | Venkata Pavana Kaushik |
| 13 | Overcoming the Failure by recovery Algorithm by assigning available slave node to Active Master. | Varun Kumar |
| 14 | Execution time estimation Of Matrix Multiplication in case of only Master doing the job | Varun Kumar |
| 15 | Execution time estimation in case of distributed approach without any failure. | Varun Kumar |
| 16 | Execution time estimation in case of distributed approch with failure | Varun Kumar |
| 17 | Power Consumption Estimation of Master and Slave without distributed Computation. | Varun Kumar |
| 18 | Power Consumption Estimation of Master and Slave with distributed Computation. | Varun Kumar |
| 19 | Demo | All Members |
| 20 | Documentation | All members |

TABLE IV: Division of work.

## VIII. Acknowledgment

We would like to express our sincere thanks to Prof Ayan Banerjee for teaching us the Mobile Computing Course. Prof Ayan has designed the course in such a way where the projects are comparable to our real life situations. By implementing this project, we understand a new way to use mobile devices for using computation, how to design a master-slave architecture and Distributed computing.

## References

[1] https://developers.google.com/nearby
[2] https://developers.google.com/nearby/connections/overview
[3] https://developer.android.com/training/location
[4] https://youtu.be/DXkcAc-0ef0