CSE 535: Mobile Offloading Project

Vageeshan Mankala[#1], Harsh Manral[#2], Hanuman Reddy Solleti[#3], Kavin Raj Aadhavan Chellamuthu[#4]
Arizona State University

Email: [#1]vmankala@asu.edu, [#2]hmanral@asu.edu, [#3]hsolleti@asu.edu, [#4]kaadhava@asu.edu

ABSTRACT

The motive of this project is to develop a distributed computing system using mobile devices. Two separate Android applications have been developed to demonstrate mobile offloading in a Master-Slave architecture. The master device scans for available slave devices and sends a notification to participating slave devices to connect and send their location and battery information. The connection between the devices is achieved with Bluetooth. The master distributes the task of matrix multiplication by assigning and sending separate rows to the slave devices and once it receives the result back, it compiles it to generates the final output. Multiple features have been added to this process to enhance it and are also discussed later in the paper.

Author Keywords

Bluetooth; Mobile Offloading; GPS; Matrix Multiplication

INTRODUCTION

One of the biggest barriers to running heavy computation on mobile devices is the high-power consumption/battery drop along with the limited processing power of mobile devices. To overcome these obstacles, the work can be distributed to different devices in the network, which barely increases the total time taken for computation but in turn ensures no single device loses too much battery or computational resources. In our application, we demonstrate the above concept of offloading using Bluetooth, to perform matrix multiplication.

PROJECT SETUP – PERMISSIONS AND REQUIREMENTS

The application requires the permissions listed below:

- ACCESS FINE LOCATION
- ACCESS COARSE LOCATION
- ACCESS_BACKGROUND_LOCATION
- BLUETOOTH_ADMIN
- BLUETOOTH_ADVERTISE
- BLUETOOTH_CONNECT
- BLUETOOTH_SCAN
- BLUETOOTH

Minimum Android API requirement – API 27

IMPLEMENTATION AND FEATURES

1. Task allotment priority

During matrix offloading, we have made sure the master sends the rows to slave devices based on a priority order calculated using a score metric consisting of the slave battery levels and location

Score = (Battery Level/100) - (Distance/0.2)

The device with the highest score always gets the first priority if it is free (not already calculating a previously given task)

2. Task reallotment if slave gets disconnected

If a row is sent to a slave device and isn't recovered within 5 seconds – it is marked as 'not sent' and reallotted to the slave devices based on the priority order mentioned earlier.

3. Rejection of offloading by slave

At initial connection, the master device asks the slave if it wants to participate in offloading and share location and battery data. If the slave device denies, the Bluetooth socket is closed and the message that the slave denied offloading is toasted at the master screen.

4. Battery and location monitoring at master and slave nodes

Added logic to ensure that no device which is more than 200 meters away or has a battery level lower than 30% is allowed to make a connection for offloading with the master. Location and battery stats are sent by the slave devices on first connection, and later only when requested by the master device.

5. Power consumption calculation:

Used an external application called 'AccuBattery' on all devices to measure approximate power usage in mAh

APPLICATION INTERFACE

Fig. Master app UI:

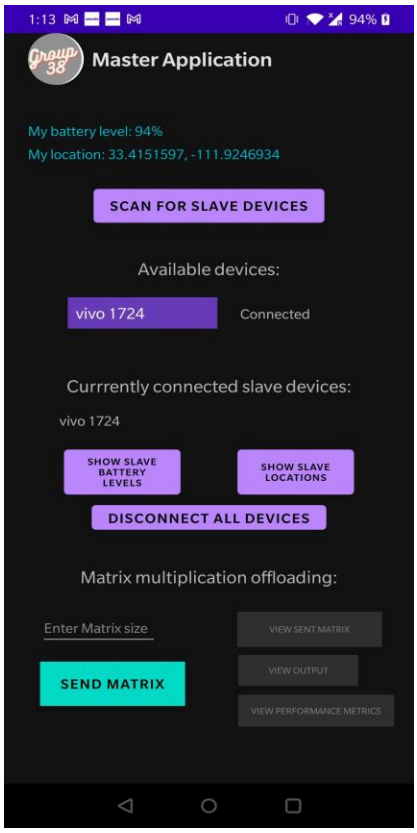

Fig. Slave app UI:

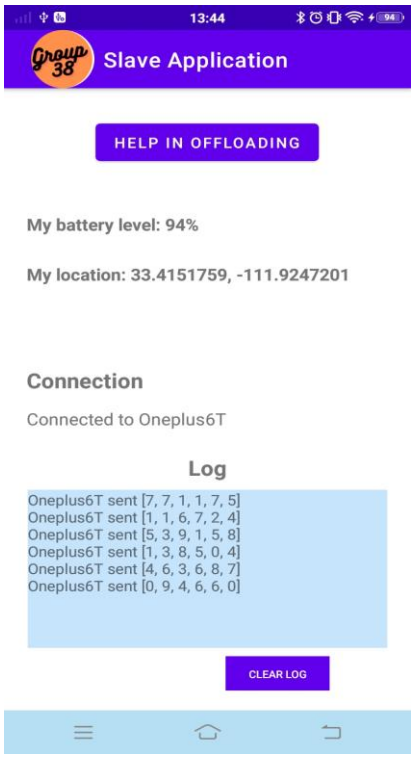

Fig. Master device sending notification to slave for offloading, and battery and location data consent:

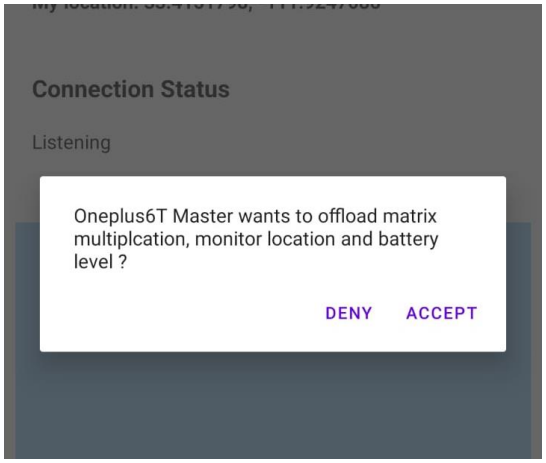

Fig. If the slave denies offloading:

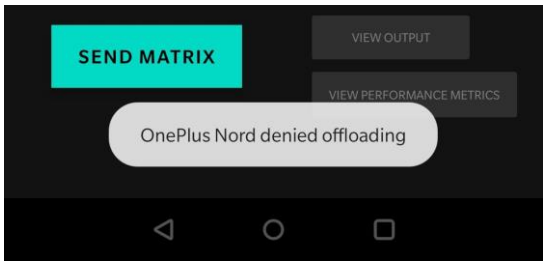

Fig. Master device with all connected slave battery data (similar for location)

COMPLETION OF TASKS

*VM - Vageeshan, HM - Harsh, KR - Kavin, HS - Hanuman

1	Develop a Master mobile application that can be used to start a program that collects battery levels from mobile phones and lists it in a file	VM, HM
2	Develop a service discovery application, that sends queries to available mobile phones in proximity through Bluetooth or WiFi to request participation	VM, HM
3	Develop a dispatcher application that choses a mobile phone among the ones that accepted the request based on matching some requirements. Requirements can be minimum battery level, and location proximity	VM, HM
4	Based on the chosen set, send requests to start battery monitoring application on the slave side	VM, HM
5	Develop a slave application that can receive a request from a master through Bluetooth or Wi-Fi	KR, HS
6	The application can also monitor the battery level and current location and send it back to the master if the user decides to consent	VM, HM
7	The application can then run a code snippet in the slave end to start the periodic monitoring application	KR, HS
8	Using this framework, solve the problem of distributed Matrix Multiplication a. The master instantiates the Matrix	VM, HM
9	b. Sends parts of the Matrix to the Slave	VM, HM
10	c. Slave computes on the Matrix	VM, HM
11	d. Sends data back to the Master	VM, HM
12	e. Master combines and shows the result	VM, HM
13	Write a failure recovery algorithm in the Master, where if a slave fails the master can reassign immediately to another available slave node	VM, HM
14	Estimate execution time of the Matrix Multiplication if done only on the Master	VM, HM
15	Estimate execution time If done using the distributed approach with no failure	KR, HS
16	Estimate execution time if done using the distributed approach with failure	KR, HS
17	Estimate power consumption of Master and Slave nodes without distributed computation	KR, HS
18	Estimate power consumption of Master and Slave nodes with distributed computation	KR, HS

LIMITATIONS

1. Upper limit on the size of the matrix the master is allowed to send, due to Bluetooth serial data size limits (maximum 990 packet size using TCP)

2. As we cannot send huge matrices, determining and comparing the actual power consumption statistics is hard with small matrices which take time in order of nanoseconds to complete

3. Devices must be in close proximity to use Bluetooth

4. Currently the application just creates a random matrix, as it is for demonstration purposes, but it can be changed to a user input matrix

5. Currently every time a row of the first matrix (A) is sent and assigned to a slave device, the entire second matrix (B) is sent along with it. We can update it such that matrix B is just sent once

RESULTS

Time taken with offloading (ns)=
682880729
Time taken without offloading (ns)=
17500

Fig. Results from the run with no failure

Time taken with offloading (ns)=
5160663435
Time taken without offloading (ns)=
24480

Fig. Results from the run with slave failure

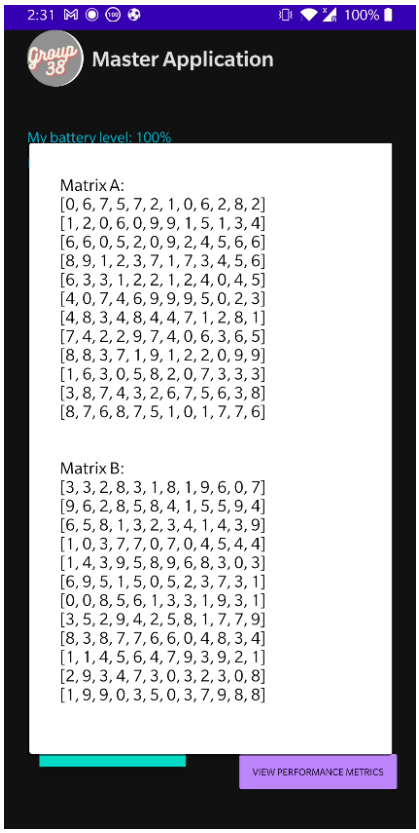

Fig. View sent matrix on the master

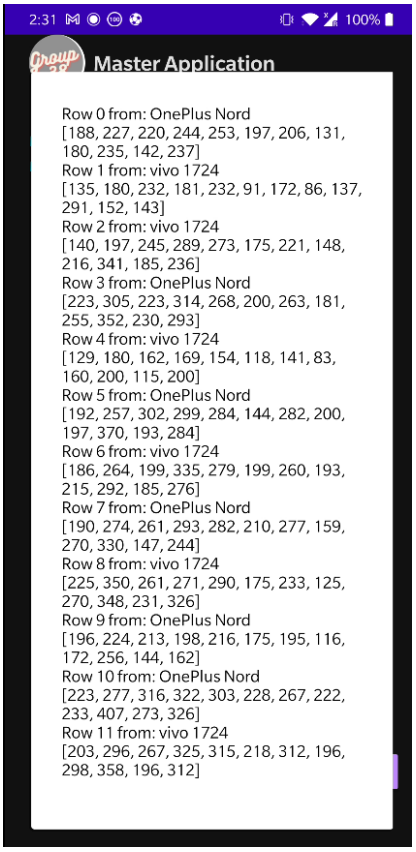

Fig. View received matrix on the master

Performance and power related metrics:

Execution time of the Matrix Multiplication if done only on the Master – 17500 ns in run with no failure, and 24480 ns in run with failure of node (note that the failure has no effect and both values are in the same ballpark)

Execution time If done using the distributed approach with no failure – 68288 * 10^4 ns

Execution time if done using the distributed approach with failure – 51606 * 10^5 ns

Power consumption of Master node without distributed computation (using AccuBattery) – ~4.4 mAh

Power consumption of Master and Slave nodes with distributed computation (using AccuBattery) - ~4mAh at Slave 1, and ~5mAh at slave 2, ~8.8mAh at master

CONCLUSION

With very little increase in computation time, by offloading work - we can reduce power consumption on just a single device and instead have meagre amounts of battery drop on all devices participating in the offloading, which is beneficial for mobile devices. Moreover, we are saving resources on the master and making it relatively free in case it needs to multitask.

ACKNOWLEDGMENT

We would like to take this opportunity to thank Professor Ayan Banerjee for enabling us to work on this project. Several concepts became evident to us during the process of implementing all the needs of this application, substantially steepening our learning curve in a short amount of time.

REFERENCES

1. Ko, J., Choi, YJ. & Paul, R. Computation offloading technique for energy efficiency of smart devices. J Cloud Comp 10, 44 (2021). https://doi.org/10.1186/s13677-021-00260-8
2. Bluetooth socket, maxReceiveSize(). https://developer.android.com/reference/android/bluetooth/BluetoothSocket#getMaxReceivePacketSize()
3. BluetoothChat sample code by Google. https://android.googlesource.com/platform/development/+/eclair-passion-release/samples/BluetoothChat/src/com/example/android/BluetoothChat/BluetoothChatService.java
4. Serializable interface in Android. https://developer.android.com/reference/java/io/Serializable
5. AccuBattery App. https://play.google.com/store/apps/details?id=com.digibites.accubattery&hl=en_US&gl=U