Mobile offloading project

CSE 535: Mobile Computing

Dhairya Bhatnagar
Arizona State University, Tempe, US
dbhatna1@asu.edu

Shiva Kumar Chintham
Arizona State University, Tempe, US
schinth5@asu.edu

Pavan Kalyan Reddy Thota
Arizona State University, Tempe, US
pthota3@asd.edu

Venkat Sai Krishna Kowkuntla
Arizona State University, Tempe, US
vkowkunt@asu.edu

ABSTRACT--- In this project, we develop a distributed computing infrastructure using mobile phones connected via Bluetooth, Our design comprises of a master mobile application installed in one phone and Slave mobile application installed in multiple phones which performs distributed computing and we implemented the task of Matrix Multiplication to demonstrate it. We get the output from the slave applications and recombine to show the result. Additionally, we evaluated the performance with and without offloading the Multiplication task.

I. INTRODUCTION

In the past decade, we have seen computing technology expand from desktops and mainframes to a wide range of mobile and embedded devices like mobile phones, autonomous robots, etc. Many of these devices have constraints of limited resources. For example, mobile phones are battery-powered which are further constrained by size. Environmental sensors are small in size, have slow processors, and small amounts of storage, and very little data security. Most of these devices use wireless networks and their bandwidths are lower than wired networks.

Meanwhile, as user demands increase and become more sophisticated complex computations are performed on these systems—for example, video processing on mobile phones and object recognition. Thus, there is an increasing gap between the demand for complex computations and the availability of limited resources. These restrictions can be addressed by mobile computational offloading, with the growth of technologies like edge computing this form of distributed computational offloading might become the next big thing.

Distributed computing is a model in which components of a software system are shared among multiple devices to improve efficiency and performance. We perform a part distributed computing task i.e. “Matrix Multiplication” on different slave mobile applications and recombine all the parts to form the resultant matrix in the master mobile application. We monitored battery and power consumption when done with and without the distributed approach.

With our project, we plan to show the reduction in execution time and process computational load when using a distributed computing infrastructure when compared to a system with non- distributed computation. The task which will be performed here is 4X4 matrix multiplication.

II. Author Keywords

Mobile offloading; performance analysis; Distributed computation; matrix multiplication; Non- distributed computation

III. Project Setup

The architecture consists of a single Master mobile device and multiple Slave Mobile devices.

1)Master Node
Master Android Device
Model: One Plus 7 Pro
OS: Android 8.0

2) Slave Node

Slave Node 1
Model: One Plus 5T
OS: Android 7.0

Slave Node 2
Model: Motorola E2
OS: Android 5.0

Slave Node 3
Model: One Plus 7
OS: Android 8.0

Slave nodes and Master nodes are set up such that they communicate via Bluetooth socket. In this architecture Master acts as a Server using Bluetooth Server Socket, and all the Slaves acts as Clients communicating with Bluetooth Client Socket.

We install Slave applications on Slave Nodes and Master applications on the Master Node prior to the initial connection via Bluetooth.

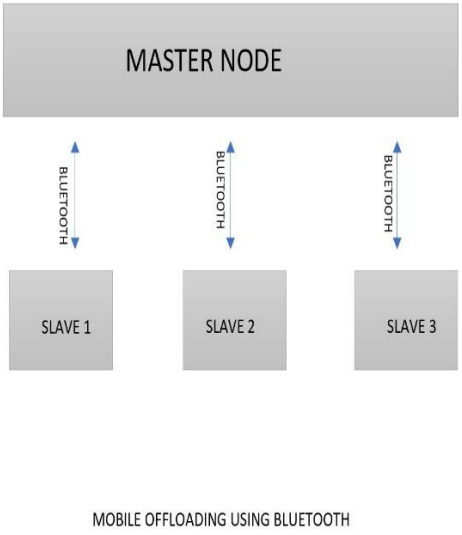


Figure 1: Architecture of Mobile Offloading.

The Communication channel here is bidirectional. Communication between the master and slave nodes will be as a JSON data string sent as a byte array.

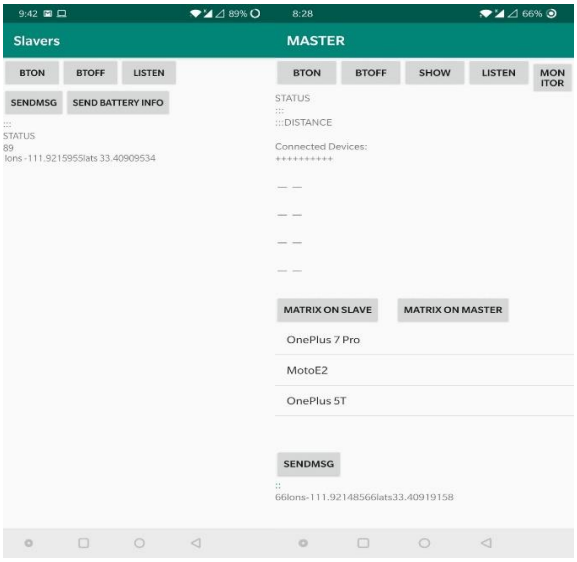
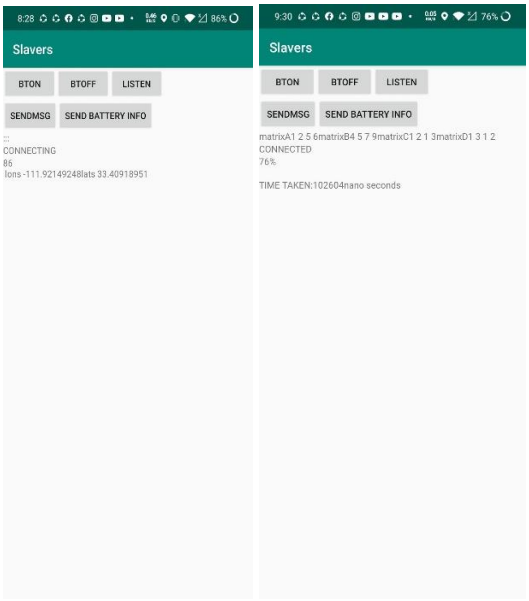


Figure 2: Slave Application 'Slavers' and client application 'MASTER'.

IV. Implementation

1. The Slave node is open on the Bluetooth channel for the connection. The Master sends a Bluetooth request to Slave. Then, the Slave accepts the request and gets connected.



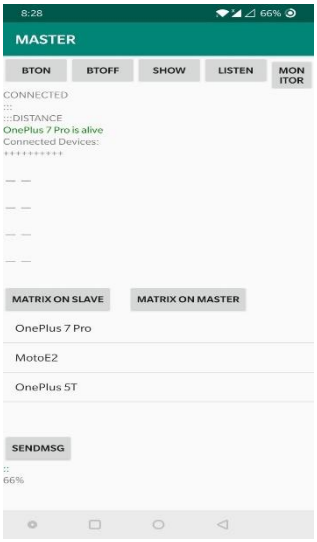


Figure 3: Establishing successful of Master and Slavers.

2. As soon as a new connection is made, the slaves send battery information to the master and the master evaluates according to a criterion and sends back a request of distributed computing to the slaves that met the criterion.

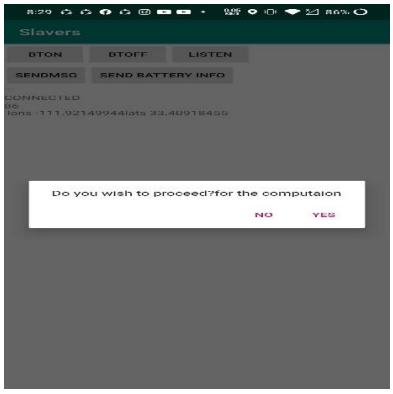


Figure 4: Request for distributed computing.

3. The devices respond to the request and send their details to the master. The details received by the master node include battery levels and GPS Coordinates. We have a selection criterion: if the distance between master and slave is less than 100m and battery level greater than 25%, the Master selects that device as a slave and sends a request to all slaves which satisfy the constraints to participate in the distributed computing.

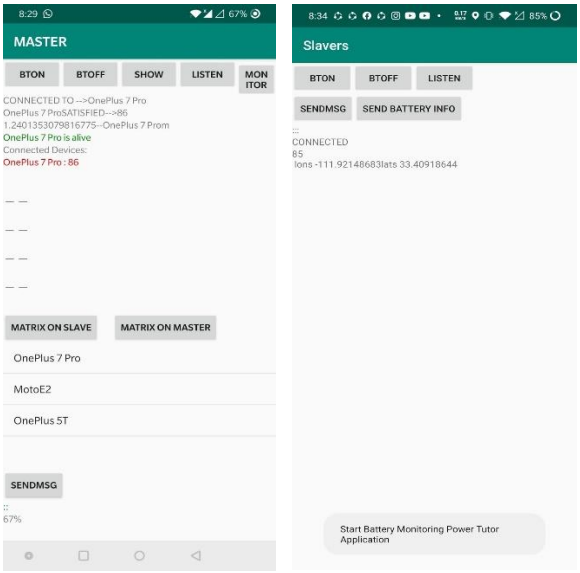


Figure 5: Start of battery monitoring.

4. The Master can check the battery status of slaves at any time. The Master writes the battery details of all the slave nodes in a file “battery.txt”.

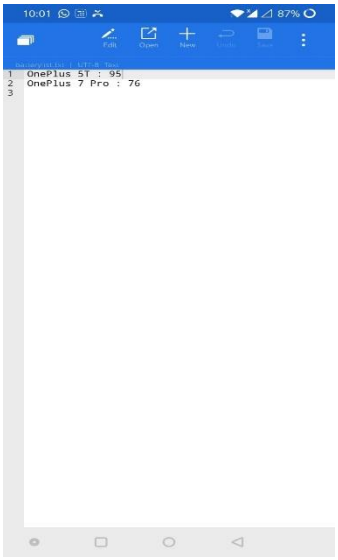


Figure 6: Maintaining the battery.txt

5. Master will periodically monitor the slave node presence. In our project, the master checks the status of the slave every 3 seconds. This check is a way to ensure if the slave is dead or alive.

Figure 7: Periodic Monitoring

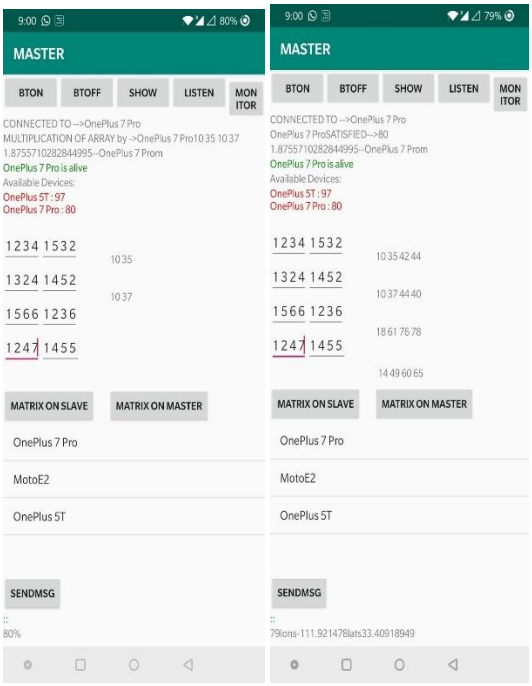


Figure 8: Final Result after Matrix Multiplication.

6. Distributed Computational operation:

   Here we are performing Matrix multiplication. We take the input from Master Application.

   We performed 4X4 matrix multiplication. Master initiates the MATRIX and each matrix is divided into 4 parts of size 2X2 as shown below.

   P1 P2 Q1 Q2

   P3 P4. Q3 Q4

   We send the parts of the matrix to different slaves and perform individual computations based on the data sent to them by the master as shown below.

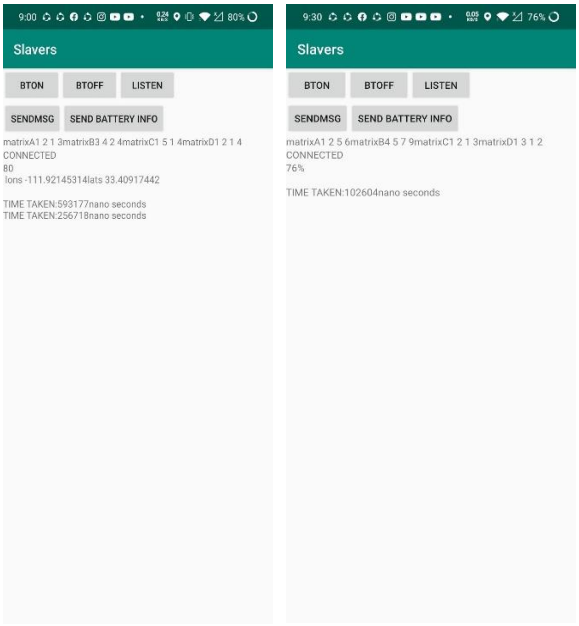
   [P1, P2, Q1, Q3] On slave 1;

   [P1, P2, Q2, Q4] On slave 2;

   [P3, P4, Q1, Q3] On slave 3;

   [P3, P4, Q2, Q4] On slave 4

Time on Slaves:



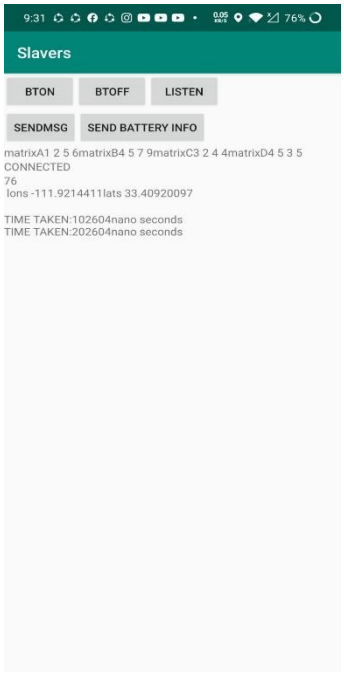


Figure 9: Execution time on slavers.

The execution time required for the matrix multiplication if it is done on the Master.

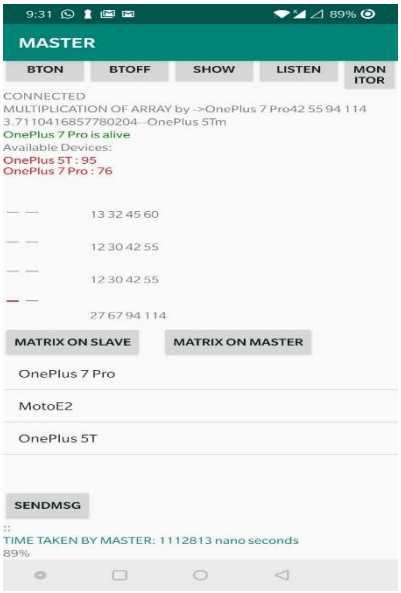


Figure 10: Execution time on Master.

7. Failure Recovery:

If a slave is dead while receiving data from the master, continuous periodic monitoring feature of the master mobile application will allow the master to know if the slave is alive or dead in a short span and to recover from this failure, we send the same data to another slave that is active.

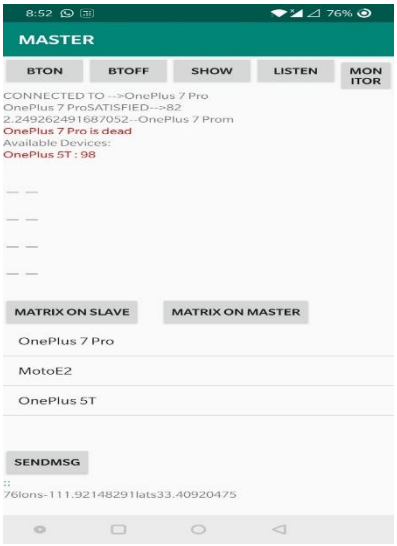


Figure 11: Failure recovery in action.

V. Observations

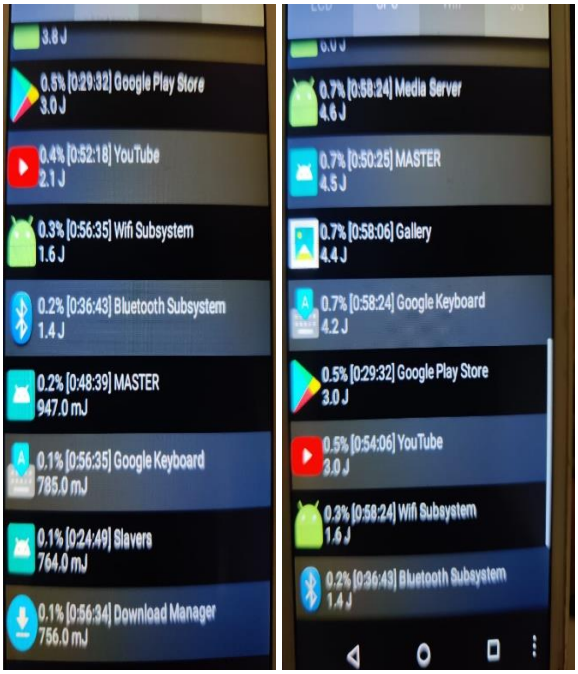


Figure 12: Observing the power consumption.

Observations	Value
Total Execution time (Master only)	1112813 nanoseconds.
Total Execution time (distributed approach) no failure	115508 nanoseconds
Total Execution time (distributed approach) failure condition	150124 nanoseconds
Total Power consumption without distributed approach	4.5 Joule
Total Power consumption with distributed approach	1.5 Joule

VII. Limitations

Distributed Computing Infrastructure solves a lot of problems that are imminent as technology keeps evolving but still being a comparatively newer technology there are certain limitations in its working. One of the major ones is the security aspect while using a distributed system speeds up the computation and reduces stress on a single system, it still involves passing information to multiple systems that might compromise security. Furthermore, distributed systems are difficult to troubleshoot as it's difficult to identify the point at which the fault occurred. The initial setup and infrastructure costs are also higher. Another major advantage of distributed computation is the scalability, in our case that would be adding more slaves to the architecture to improve performance, but this also comes at the cost of system complexity. We plan to study these issues and create a framework that could handle these issues for our future work.

VIII. Conclusion

In this project, we developed a distributed computing infrastructure which performed Matrix Multiplication. Our results displayed a sharp reduction is execution time while using the distributed infrastructure and also lead to reduced power consumption. We also showed that our infrastructure can recover from Slave device failure scenarios.

IX. References

[1] Bertocco, Matteo & Ferraris, Franco & Offelli, Carlo & Parvis, Marco. (1998). A client-server architecture for distributed measurement systems. Instrumentation and Measurement, IEEE Transactions on. 47. 1143 - 1148. 10.1109/19.746572.