Review Article

An Improved Model for Node Discovery using Election Algorithm in Wireless Sensor Network.

ABSTRACT

Mobile wireless devices are constrained based on resources. Computation offloading provides a unique method for taking advantage of multiple mobile wireless devices to save energy and increase performance of these devices. In order to build a robust serious solution for a mobile wireless sensor network. There must be a scheme to ensure that the resources of the mobile wireless sensor network is managed adequately. However, computational offloading scheme was proposed by researchers. But this solution was dependent on a super node which manages the offloading process and is required to be online on the network at all time which consume time. The objectives of this study is to create a "broadcast and receive" table that can access a given node at a particular time interval without the need of having a super node online all the time. This method ensures that as nodes enter the network, they announce their resources which is then saved in the table. This information is updated at a given time interval by a monitoring service. When a node is to be removed from the network, the details of the resources of the node is removed from the table. The focus of the study is to evaluate the resource aware of node discovery in wireless sensor network.

Keywords: WSN, MCC, Computational offloading, Node discovery, Device profiler, "Broadcast and Receive",

1.0 INTRODUCTION

Despite the evolution and enhancements that mobile wireless devices have experienced, they are still considered as limited computing devices. Therefore, Mobile Cloud Computing (MCC) integrates mobile computing and Cloud Computing (CC) in order to extend capabilities of mobile wireless devices using offloading techniques. Computation offloading tackles limitations of Smart Mobile wireless Devices (SMWDs) such as limited battery lifetime, limited processing capabilities, and limited storage capacity by offloading the execution and workload to other rich systems with better performance and resources.

In WSN, node discovery assumes importance for effective communications. Since WSN supports the dynamic addition of new nodes and the departure of existing nodes, the neighbors of a sensor node are not static, they may change dynamically, and it is essential to discover new nodes. There are concepts of initial neighbor (super peer) discovery and continuous neighbor discovery. Figure 1 presents the node discovery process through the control/monitoring center.

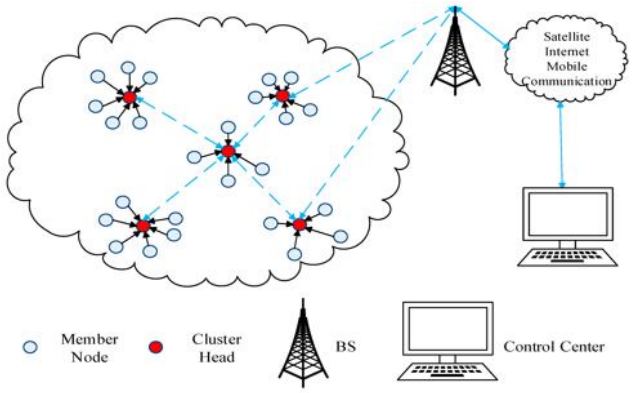

Fig 1: Wireless Sensor Network (WSN) physical structure

Although many advances in technology, mobile wireless devices will always be resource limited, as restrictions on weight, size, battery life, and heat dissipation impose limitations on computational resources and make mobile devices more resource constrained than their non-mobile counterparts. Therefore, mobile wireless devices still fall short to execute many rich media and data analysis applications that require heavy computation, and often also have (near) real-time constraints such as Augmented Reality (AR). This problem is the fact that the super peer is constantly having to monitor the network to discover peers and also serves as a mediator between different peers for any offloading task. This makes the system drain energy and computing resources.

This research proposes an improved model for node discovery using a broadcast-and-receive technique which eliminates the use of the super node and transfers the discovery process between network nodes. This proposed model will reduce energy and computational resource consumption (consumed by the super node) and hence improve the computational offloading process. The composition of this paper is as follows. Section 2 reviews related research that examines computational offloading methods to reduce the energy consumption of sensor nodes. Section 3 proposes node discovery model to reduce the energy consumption of sensor nodes. In Section 4, the performance and analysis of the proposed model is verified through simulation. Section 5 provides conclusion and future tasks

2.0 RELATED WORKS

WSN research can be classified into different methods for improving the performance of the WSN system through performance improvement of the configuration and the sensor nodes, which are the major components and methods for improving performance by reducing energy consumption during node detection, data transmission, routing.

Wireless sensor networks are very useful in applications where direct interaction with humans is difficult, (Miriam, Ernesto, & Mario, 2016), remarked that reconfiguration is easy to implement but requires more network resources with a high energy cost. (Miriam, Ernesto, & Mario, 2016), analyzed that the formation is either event detection based or not, and network backbone is formed or not and also remarked that distributed solutions are preferred over centralized ones.

In this paper, a method is proposed to improve the lifetime of a WSN through improvements in node discovery and cluster head election with the basics on aware of resources. Accordingly, different methods and models are compared and analyzed for a typical WSN. WSN routing protocols can be divided into flat-based protocols, hierarchy-based protocols, and location-based protocols. The features of each configuration method have various disadvantages and disadvantages

(Kinsley, et al., 2020), centered most challenges of WSNs on the inadequate power resources. Nevertheless, in designing either software or hardware, it is important to cautiously contemplate on the issues of resourceful energy use, which could be extra energy for computation, manipulation or filtering. To overcome this issue, it could be suitable to turn off a subdivision node in order to preserve and conserve energy based on the application, moving a computational task from one device to another is not a trivial endeavor.

Each system defines particular opportunistic criteria to estimate the effort to offload. A mobile wireless device that uses an offloading system, detects opportunities to offload when an application is executed and is in a networked environment (Gupta, Vahid Dastjerdi, Ghosh, & Buyya, 2017). Thus, the augmentation of the mobile resources with external infrastructure is temporal as long as the criteria are fulfilled.

Computational offloading is the opportunistic process that relies on external infrastructure to execute a computational task outsourced by a low-power device. (Flores, et al., 2017), in this process, the device is granted with the decision logic to detect resource intensive tasks, such that in the presence of network communication, the device can estimate where the processing of the task will require less computational effort (internal or external), which saves the device's energy. By outsourcing a task, overall the mobile wireless device consumes less resources, and in some cases, even the response time of the application is accelerated (Gordon, et al., 2012) Computational offloading is an integral part of mobile cloud computing which has attracted so much attention in recent times since it is a way of saving energy in mobile wireless devices by sending an intensive task to a remote server for execution. Mobile offloading is a promising technique to aid the constrained resources of a mobile wireless device. By offloading a computational task, a device can save energy and increase the performance of the mobile wireless network

(Kim, et al., 2021), proposed routing protocol similar to LEACH in order to minimize the energy consumption based on node's physical location and cluster information. In addition, though energy consumption is evenly distributed among nodes to improve network life and performance. However, looking at the energy consumption distribution of each node in their proposed protocol, as the distance from the base station increases, the energy consumption of the sensor node adjacent to the base station increases.

SPIN protocol was proposed by (Shivalingagowdal, Jayasree, & Sah, 2020) , versatile in communication and data-centric routing protocol, the node interacts with one another through utilizing meta-data before transmitting the actual information. Therefore, it prevails with regards to staying away from the blind utilization of resources and avoid implosion and overlap issue. Blind forwarding is the problem that occurs due to the transmission in the direction away from the sink node causing total energy consumption. . However, it cannot

overcome the problem of blind forwarding and data inaccessibility. In contrast, the proposed system implements an incentive approach based on credit to motivate the mobile users to participate actively and accumulate more credit. To achieve this, the proposed hybrid system extrapolates characteristics of a super-node based system in order to foster an offloading system that can be sustained by a community on the long run.

3.0 METHODOLOGY

This research requires a node cluster module, responsible for managing the nodes that makes up the entire WSN network. Since the WSN network comprises of nodes that interact and exchange information in other to perform the designed task of the network, it is important to manage them in such a way as to ensure that the system resources are not unduly exhausted. The node cluster module performs two main functions:

3.1 WSN Resource Determination

One of the jobs of the node cluster module is to determine the resource of the wireless sensor node. The nodes are required to announce resource upon entry of the network. The node cluster module reads these resources and store them in a resource table. The resource table is not maintained in memory so as to reduce the energy and space requirements of maintaining such table. The table is only brought in memory when required by the node cluster module. The table holds information such as the computational power, energy level and memory resource.

Table 1: Example of the Resource table used by the node cluster module

Node ID	Computational Power	Storage Size	Energy Level	Status
1	1.72GHz	500mb	2.7Ah	active
2	2.2GHz	1000mb	2.9Ah	active

3.2 WSN Safe state determination

The job of the node cluster module is to determine the safe state configuration of the WSN, the node cluster module uses this safe state to create a safe state vector for the WSN which will serve as possible legal actions that the WSN can perform in the WSN network. This vector will hold possible and allowable actions which are declared legal for the WSN on the network. The safe state vector is implemented using the dictionary data structure so that allowed actions are assigned a name for identification purposes, hence for example, say we have a node called “ND1” on the network and we also go further to assume that the node has announced its resources and that has been stored in the resource table (the safe state vector generation and resource announcement will all happen at once as one step and not as separate steps as discussed here), the node cluster module will go further to generate a safe state vector for the node using a dictionary data structure as below:

{“ND1”: [“physical layer access”, “MAC access”]}

That way, the vector holds data for the allowed level of access allowed for that node by the network as determined by the node cluster module. That way, the node cluster module can

keep track of the resources and the safe state vector information for a particular node and produce them as needed by the proposed solution.

3.3 "Broadcast and Receive sub" module

For the node cluster module to properly perform its task with the most minimum amount of resources, it uses a sub module called the "Broadcast and receive" module. The broadcast and receive sub module is the module responsible for the management of the network resource table discussed in this section. the broadcast and receive sub module can assume different states at different times. The states that the broadcast and receive sub module can assume are four (4), which are:

a. On,
b. Off,
c. Idle, and
d. Discovery.

Pseudocode 1 describes the algorithm for the "Broadcast and Receive" sub module.

Pseudocode 1

```
Procedure Broadcast_Receive()
    Default_State = OFF_STATE;
    Network_State = OFF;
    If Network_State == ON then
        Change Default_State to DISCOVERY_STATE
        If network dictected == true then
            Discover devices on the network , Triger Device_Profiler and change
            Default_State to ON_STATE;
            If Default_State == ON_STATE then
                Broadcast device status and receive node resource information
                from Network_Resource_Table;
            Else if Default_State == IDLE_STATE then
                Go into sleep mode and wait for STATE_TRIGER_TIMER;
        End if
    End if
End Procedure
```

The node cluster module works hand in hand with the election algorithm module. This is because, the election algorithm helps to determine which node in the node cluster is to be made the node cluster head, and hence we discuss the election algorithm module in the next section of this research work.

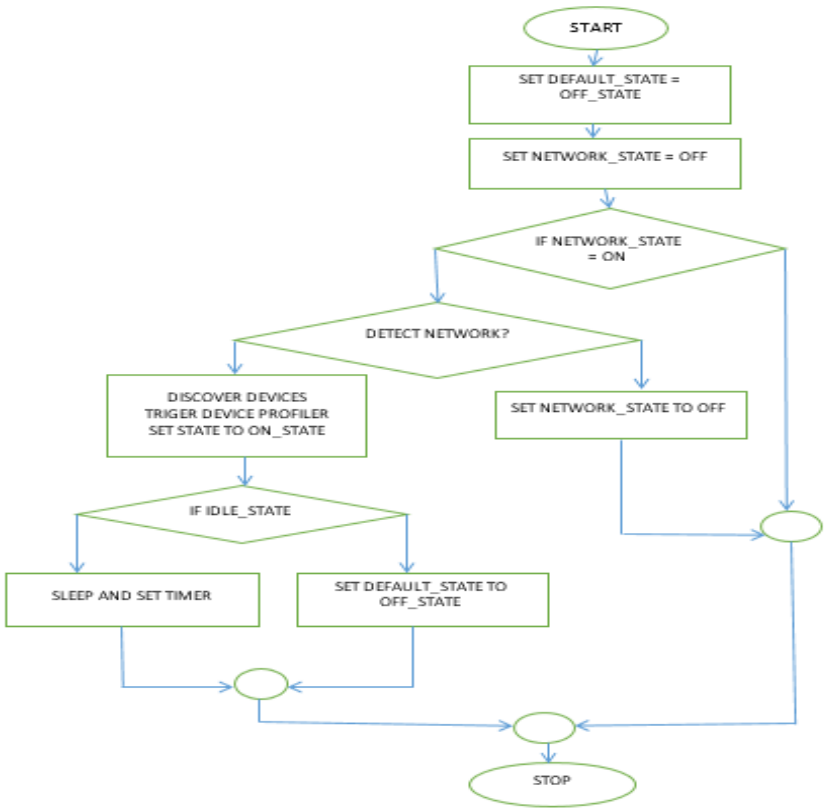

Figure 2: Flowchart of the “Broadcast &Receive” Sub module

3.4 Design of the Election Algorithm Module

The election algorithm module is responsible for the profiling of nodes on the network. The device profiler of the election algorithm module is triggered by the node cluster module when a new node or device enters the network or when a specified time interval that has been set before hand is expired. Another scenario when the device profiler could be triggered is when a node is about to leave the network. Once any of the above mentioned events takes place, the device profiler of the election algorithm module starts collecting resource status information from the resource table like the battery level of nodes on the network, the computational resource of nodes on the network and storage parameters of nodes on the network. The information so gathered is used by the election algorithm module to decide which node in the node cluster is to be made the node cluster head.

The election module algorithm is given in Pseudocode 2 for the device profiler.The functioning of the device profiler in the election algorithm is heavily dependent on the events as explained earlier on the broadcast and receive module, the getDeviceStatus() module serves as the module for performing device status rating.

Pseudocode 2

```
Procedure deivceProf()
    Set Devive_Profiler == SLEEP;
    If New_Device_Detection == TRUE then
        Set Device_Profiler == AWAKE;
        Call getDeviceStatus();
        Set Device_Profiler == SLEEP;
    Else IfTIME_INT == EXPIRED then
        Set Device_Profiler == AWAKE;
        Call getDeviceStatus();
        Set Device_Profiler == SLEEP;
    Else If DEVICE_EXIT == TRUE then
        Set Device_Profiler == AWAKE;
        Call getDeviceStatus();
        Set Device_Profiler == SLEEP;
    Else
        Set Device_Profiler = SLEEP;
    End if
End Procedure
```

The device profiler in figure 3 is very important to the whole proposed framework as it is required to gather device status information from device nodes in the network from the network resource table. This information is very important to the decision of which node is to be the network node cluster head, that is capable of handling computation offloading.

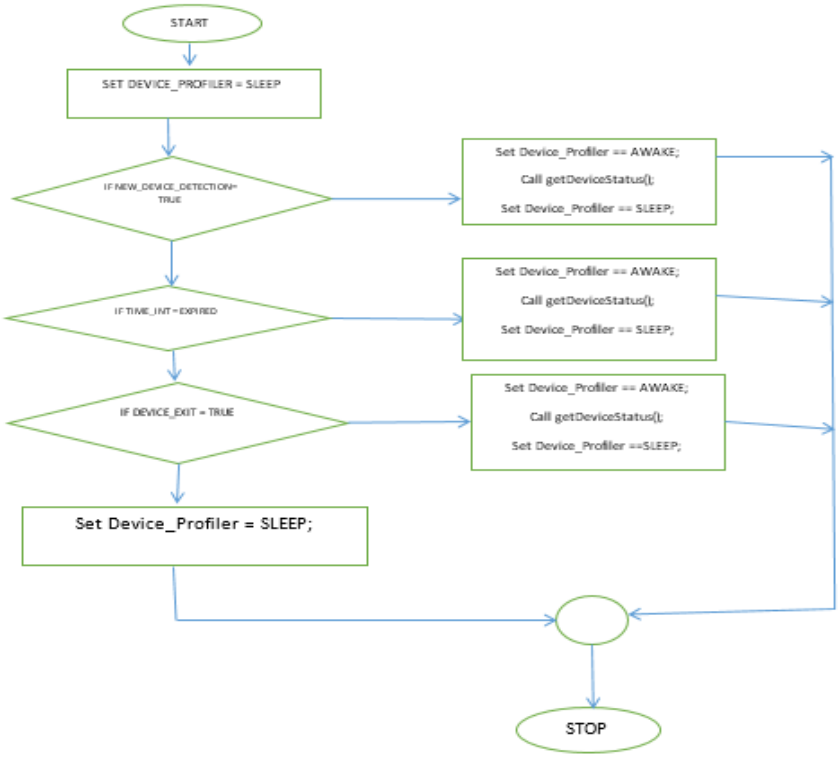

Figure 3: Flowchart of the Device profiler algorithm

In the algorithm in Pseudocode 2, the getDeviceStatus() module serves as the module for performing device status rating. This calculations help reduce the work of selecting the network node cluster head. The calculations is based on three (3) parameters, which are:

1. The computing resource of the device,
2. The number of nodes connected to the device, and
3. The energy level of the device.

The equation (1) below shows the the equation used for the device ranking.

Score(Nc,Cc,Bc) = (Nc/N) * (Cc/100) * (Bc/100) ---------- (1)

Where:

Nc = the number active connections to the node,

Cc = the computing resource processing frequency,

Bc = the battery level of the node.

So say we have a cluster node in the network which is atively connected to three (3) nodes in the network of a total connection of ten (10) nodes and has a processor of 2.0MHz and a battery level of 65%, then we will have a score of:

Score(3,2.0,65) = (3/10) * (2.0/100) * (65/100) = 0.0039,

So the cluster node has a score of 0.0039 at the time this calculation was performed by the device profiler of the election algorithm and the computed score will be used to update the network device status table by the broadcast and receive module.

Figure 4 shows the behavior of the proposed election algorithm in the event of network partitions and combinations. The figure above depicts a hypothetical scenario of use of the proposed election algorithm. The figure shows four different cluster nodes N1, N2, N3, and N4. The two different scenes are S1 and S2 and NS is used to indicate which WSN is acting as the cluster head at any particular point in time. In the above scenario, the proposed election algorithm can be used to satisfy the computational needs of a/some pairs in the system.

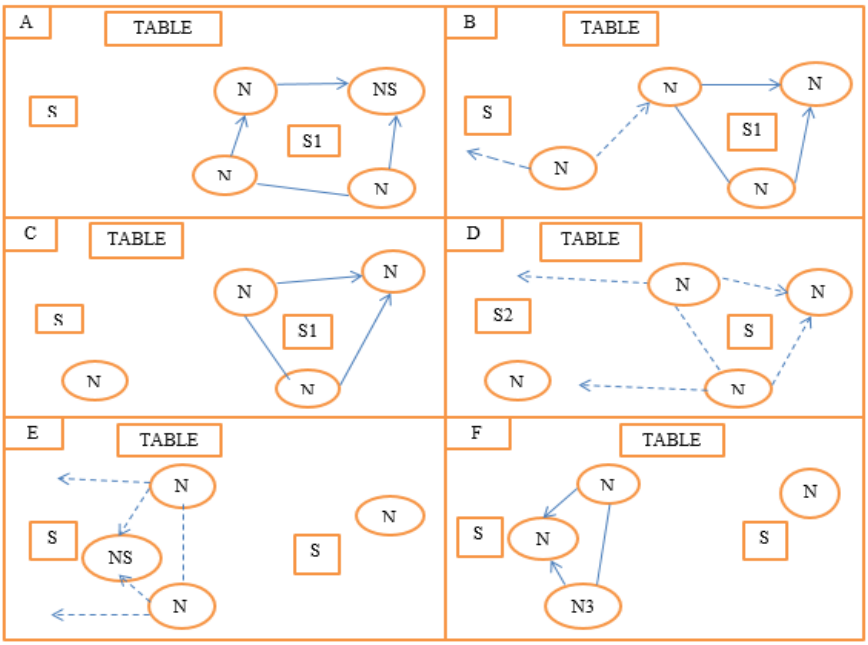

Figure 4: Election algorithm

Nodes N1, N2, N3, and N4 all enters scene S1 and broadcasts their computational requirements and make up to the broadcast and receive table. The computational requirements here means the computational needs of the pairs and the computational make up here means the computational strength, battery level and memory of a pair in the network. A device profiler in the network ranks these broadcasts and decides from the ranking the node to become the cluster head NS.

A. A node from scene S1 decides to leave the network to join the network in scene S2. Since the network in scene S2 is out of the reach of the network in scene S1, the movement triggers the broadcast and receive module to update its status to reflect the new situation of the network. The device profiler would collect the different profiles of the remaining nodes in the network and re-rank them again to determine the node that will serve as the new cluster head NS and also update the status of the network to reflect this new development.
B. At this point, the network is partitioned into two different networks scenes S1 and S2 where N1,N2,and N3 are in scene S1 and N4 is in scene S2. While N1 still remains the node cluster head in scene S1, N4 is a cluster head to itself only in scene S2. The broadcast and receive

table in scene S2 is updated to reflect this. At this point, cluster nodes N1 and N2 indicate their intention to move to scene S2 and the broadcast and receive modules prepares the network for this migration.

C. The cluster nodes that left scene S1 now joins scene S2 and hence creating a connection between them and cluster node N1. At this point, the device profiler collects the profiles of the individual nodes on the network and ranks them using the parameters already discussed. The broadcast and receive module updates the network table with this new information.

D. The computed ranking from above is then used by the broadcast and receive module to determine the node that is to be the scene cluster head NS for scene S2. After due consideration of the scores, and the computational needs of the nodes in this group, a node cluster head is determined.

3.5 Design of the Monitoring Service Module

This module acts as the control unit of the whole system, it is responsible for coordinating all of the activities of the system. The monitoring service module controls the election algorithm module, the request generation module and the residue number system module. The monitoring service module is responsible for coordinating all the activities of the system. It interacts primarily with the request generation module by interpreting actions that are to be performed for each module in the system. This module is responsible for controlling the way all other modules are fired and also determines when to put modules to sleep for a time interval. Just like the operating system of any computing system, it is the operator and coordinator of all activities and request in the proposed solution. It is the general manager and the resource manager of the system this module, just like the election algorithm module, is fired only when there is a need for action. It sleeps when no activity is required and wakes after a particular time interval. It comes alive after every time interval to ensure that all is functioning as is and goes back to sleep. It only wakes before a time interval if there is an interrupt by any of the other module requesting some action to be taken on their behalf. An interrupt can occur when a module has completed a time interval or when requested by another module. Say for example that a new node is to be introduced to the network, this will trigger the monitoring service to fire the node cluster module which in turn will trigger the election algorithm. Other than the RNS module, this is the most active module as all requests in the system goes through it and all actions are routed by it.

4.0 RESULT, ANALYSIS & DISCUSSION

A mobile client was developed and installed in the client nodes. The mobile application client peers were designed with Java and the android studio integrated development environment.

The simulator uses a network peer status log file which represents a small space in memory where the status of individual peer in the network is stored at every specified interval and completely managed by the broadcast and receive module as shown in figure 5. This log file consumes no computing resource to maintain and hence requires little or no energy at all to operate or to maintain. The log file requires a light maintainace algorithm to keep information in it up to date and relevant. This maintainace algorithm is managed by the broadcast and receive module and used by the simulator to handle updates and statistics of the nodes on the

network. The device profiler is responsible for the profiling of devices on the network. The simulator implements a device profiler which is trigered when a new device enters the network or when a specified time interval that has been set before hand is expired. Another scenerio when the device profiler could be trigered is when a divice is about to leave the network. Once any of the above mentioned events takes place, the device profiler starts collecting device status like the battery level,computational resource and storage parameters of devices on the network.

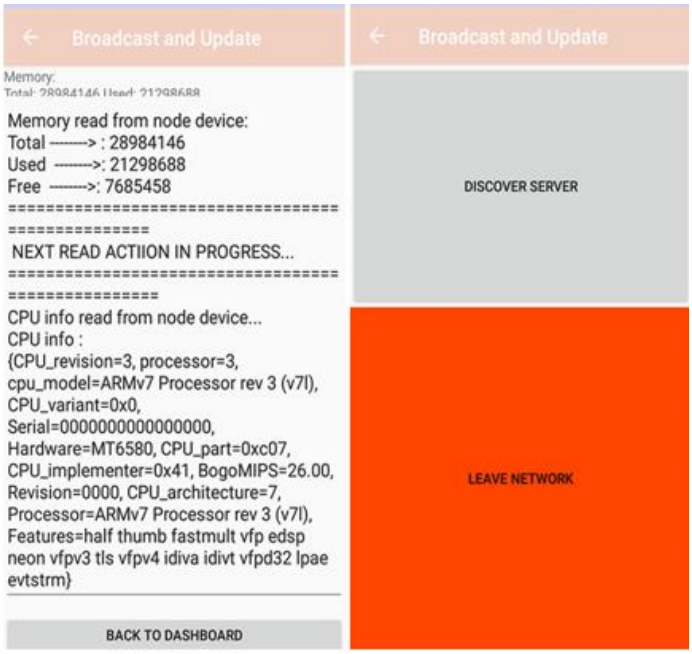

Figure 5 :The Mobile client interface

The experiment that was carried out was done to replicate the work of Huber Flores et al, this was done so as to subject the developed solution to the same scenerio in order to be able to compare the outcomes.

Evaluation and analysis on the framework based on Peer discovery performance was done

4.1 Huber Flores et al

Huber Flores et al developed a social-aware hybrid offloading system (HyMobi), which, they claimed, increased the spectrum of offloading opportunities. As a mobile device is always colocated to atleast one source of network infrastructure throughout of the day, by merging cloudlet, device-to-device and remote cloud offloading. They used ten mobile devices to carry out their experiment. The result of the social hybrid experiment is presented in tables 2& 3.

Table 2: Discovery time of Huber et al

PEER	1–3 (ms)	4–6 (ms)	7–9 (ms)	10–12 (ms)	13–15 (ms)	16–18 (ms)	19–21 (ms)	22–24 (ms)
P0	500	470	600	500	480	450	500	500
P1	1300	1500	1800	2700	2300	1200	1500	2000
P2	2500	1870	1770	1450	1980	1230	1520	1700
P3	760	1000	800	720	1060	1040	550	600
P4	2200	2320	2520	2490	2730	2620	2320	2800
P5	850	1080	720	1032	763	650	860	600
P6	2830	2230	2730	2290	2120	2980	2230	2700
P7	3301	3730	3120	3890	3530	3930	4120	4400
P8	2520	2300	2230	2590	2320	2290	2540	2000
P9	1170	1879	2290	2770	2312	2630	1980	1700

Table 3: Summary of the Huber et al experiment

PEER	CPU (Hz)	DISCOVERY TIME (MS)	BATTERY LIFE (%)	RAM (GB)
P0	Quad-core1.4	500	76	1
P1	Quad-core2.7	2000	72	3
P2	Dual-core1.2	1700	81	1
P3	Dual-core1.2	600	55	1
P4	Quad-core2.5	2800	81	3
P5	Quad-core1.5	600	61	4
P6	Quad-core1.4	2700	76	2
P7	Quad-core1.2	4400	74	1
P8	Quad-core1.8	2000	74	2
P9	Dual-core1.5	1700	81	4

Ten mobile devices were monitored based on Huber's model and generated the above result for discovery time of peers and energy values after their experiment. The graph of this results is shown in figure 6 on the discovery time for each of the ten peers.

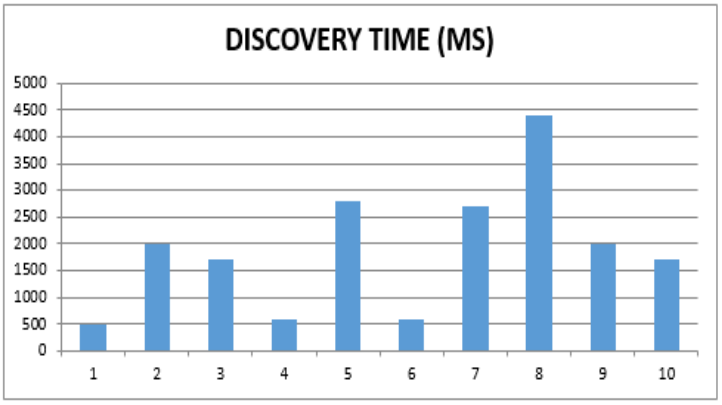

Figure 6: Discovery time in miniseconds for Huber et al

4.2 Developed Model

The developed model was also experimented on the ten devices . The result of the experiment is presented in tables 4 – 5

Table 4: Discovery time of the developed model.

PEER	1–3 (ms)	4–6 (ms)	7–9 (ms)	10–12 (ms)	13–15 (ms)	16–18 (ms)	19–21 (ms)	22–24 (ms)
P0	900	1200	2200	1000	1500	700	1300	1800
P1	1300	1000	800	700	1300	900	500	500
P2	2500	1870	1770	1450	1980	1230	1520	1870
P3	700	900	500	1020	760	840	550	500
P4	1200	1320	1520	1490	1730	1620	1320	1490
P5	750	980	620	932	663	450	760	550
P6	1830	1230	1730	1290	1120	1980	2230	1820
P7	2301	1730	2120	1890	1530	1930	2120	1880
P8	1520	2300	1230	1590	1320	2290	1540	1700
P9	2170	1879	1290	1770	2312	1630	1980	1770

Table 5 :Summary of the experiment result for the developed model

PEER	CPU (Hz)	DISCOVERY TIME (MS)	BATTERY LIFE (%)	RAM (GB)
P0	Quad-core1.4	1800	82	1
P1	Quad-core2.7	500	69	3
P2	Dual-core1.2	1870	76	1
P3	Dual-core1.2	500	72	1
P4	Quad-core2.5	1490	85	3
P5	Quad-core1.5	550	58	4
P6	Quad-core1.4	1820	80	2
P7	Quad-core1.2	1880	76	1
P8	Quad-core1.8	1700	70	2
P9	Dual-core1.5	1770	79	4

The result of the experiment was also presented in a graph in figures 7.The graph shows the discovery time of the results.Replicating the scenerios were done and the events for the experiment.

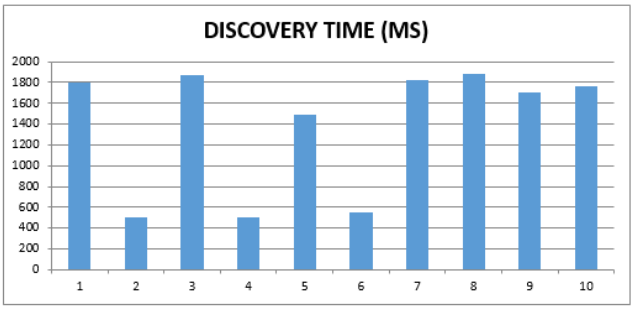

Figure 7 : Discovery time for the developed model

Figure 8 present a comparison of the two results in a graph with the developed model in blue and the model of Huber Flores et al in orange.

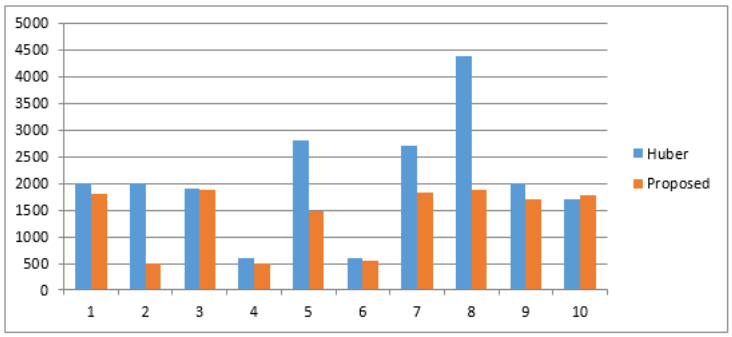

Figure 8 : Comparison of Discovery time (ms) against ten devices.

5.0 CONCLUSION

In this research work, architecture for peer discovery and peer interaction is developed in such a way as not to spend too much computing resource during this discovery and interaction process. Thus, the developed architecture requires very minimal resource for interaction and communication.The ultimate goal of this reasearch is to create communities, in which a device can lease and acquire offloading support based on very minimal resource requirements. By introducing machine learning into the network discovery process, one could be able to predict how long a network scene cold last and how best to handle a user request. This prediction approach would help to improve the discovery process of the node-server-peer and save its energy.

REFERENCES

Cuervo, E., Balasubramanian, A., Alec Wolman, D.K. C., Saroiu, S., Chandra, R., & Bahl, P. (2010). MAUI: Making smartphones last longer with code offload. *In Proceedings of the 8th international conference on Mobile systems, applications and services (MobiSys '10)* (pp. 49-62). New York, NY, USA: Association for Computing Machinery. doi:DOI:https://doi.org/10.1145/1814433.1814441

Flores, H., & Srinama, S. (2013). Adaptive code offloading for mobile clou applications: Exploiting fuzzy sets and evidences-based learnng. *MCS 2013- Proceedings of the 4th ACM Workshop on Mobile Cloud Computing and Services.*, (pp. 9-16).

Flores, H., & Srirama, S. (2013). Mobile cloud messaging supported by XMPP primitives. *In Proceeding of the fourth ACM workshop* (pp. 17-24). New York, NY, USA: Mobile cloud computing and services (MCS '13Association for Computing Machinery. doi:https://doi.org/10.1145/2482981.2482983

Flores, H., Rajesh, S., Denzil, F., Vassilis, K., Jukka, M., Sasu, T., . . . Hui, L. (2017, April). Social-aware hybrid mobile offloading. *Pervasive Mobile Computing., 36*, 25-43. doi:doi.org/10.1016/j.pmcj.2016.09.01

Gordon, Mark, J., Delaram, M., Scott, M., Chen, Z. &., & Xu. (2012). COMET: code offload by migrating execution transparently. *Proceedings of the 10th USENIX Conference on Operating Systems Design and Implementation.*, (pp. 93-106).

Gupta, H., Vahid Dastjerdi, A., Ghosh, S., & Buyya, R. (2017). iFogSim: A toolkit for modeling and simulation of resource management techniques. *In the Internet of things,ege and fog computing environments., 47*(9), 1275-1296. doi:https://doi.org/10.1002/spe.2509

Han, Bo, H., Pan, K., Sritesh, M., Madhav, S., Jianhua, S., & Aravind. (2012). Mobile Data Offloading through Opportunistic Communications and Social Participation. *Mobile Computing,, 11*, 821-834.

Kim, J., Lee, D., Hwang, J., Hong, S., Shin, D., & Shin, D. (2021). Wireless Sensor Network (WSN) Configuration Method to Increase Node Energy Efficiency through Clustering and Location Information. *Symmetry and Asymmetry in Communications Engineering, 13*(3). doi: https://doi.org/10.3390/sym13030390

Kinsley, E. U., Ituabhor, O., Silas, S. T., Rout, G. K., Akinola, S., & Ayodotun, O. (2020). Wireless sensor networks: Application and Challenges. doi:Doi:10.5772/intechopen.93660

Miriam, C. M., Ernesto, L. -M., & Mario, S. (2016). Wireless Sensor Networks Formation :Appraches and Techniques. *Journal of Sensors.*, 18 pages. doi:https://doi.org/10.1155/2016/2081902

Shivalingagowdal, C., Jayasree, P., & Sah, D. (2020). Efficient Energy and Position Aware Routing Protocol for Wireless Sensor Networks. *KSII TRANSACTION ON* INTERNET AND INFORMATION SYSTEMS, *14*(5), 1929-1950. doi:http://doi.org/10.3837/tiis2020.05.004

Dada T.O., Togun O.A., Adegbile, A. A., Olanrewaju, O.T., Idowu I. R., Adewale, F.O., Nwufoh C.V., Oluwasegun Z.P., Alabi F.A., Abdul-Kareem A.B. Design and Construction of Stand-Alone weather monitoring System Annals of Research. Journal Federal College of Animal Health and Production Technology, 2021, Vol. 1: 232-237

Sokol, K., Andrius, A., Pan, H., Richard, M., & Xinwen, Z. (2012). ThinkAir: Dynamic resource allocation and parallel execution in the cloud for mobile code offloading. Proceedings - IEEE *INFOCOM.* (pp. 945-953,). IEEE INFOCOM.

Yong, L., Ting, W., Pan, H., Depeng, J., & Sheng, C. (2014, June). Social-Aware D2D Communications: Qualitative Insights and Quantitative Analysis. *SMART*-DEVICE-TO-SMART-DEVICE, 150-158.

Panigrahi, C. R., Sarkar, J. L., Pati, B., Buyya, R., Mohapatra, P. & Majumder, A. (2021).

Mobile Cloud Computing and Wireless Sensor Networks: A review, integration architecture, and future directions. IET NETWORKS, 10 (4), pp.141-161. https://doi.org/10.1049/ntw2.12013.

Wang, L., Shao, H., Li, J., Wen, X. & Lu, Z. (2020).Optimal Multi-User Computation

Offloading Strategy for Wireless Powered Sensor Networks: IEE access, Open Access Journal, Vol. 8, 2020.Digital Object Identifier 10.1109/ACCESS.2020.2967559