International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395 -0056
Volume: 02 Issue: 07 | Oct-2015 www.irjet.net p-ISSN: 2395-0072

TASK ORIENTED & SCHEDULING ON LOW POWER CONSUMPTION IN MOBILE CLOUD COMPUTING

[*1]Ms. Arun Kumari G., [*2] Ms. Sangeetha Lakshmi G., [*3] Ms. Siva Sankari A.,

*[*1] M.Phil Research Scholar, Department of Computer Science, D.K.M. College for Women (Autonomous), Vellore, TamilNadu, India.*

*[*2,] Assisant Professor, Department of Computer Science, D.K.M. College for Women (Autonomous), Vellore, TamilNadu, India.*

*[*3]Head of the Department, Department of Computer Science, D.K.M. College for women (Autonomous), Vellore, TamilNadu, India.*

Abstract - ***As mobile computing has been developed for decades, a new model for mobile computing, namely, mobile cloud computing, emerges resulting from the manager of powerful yet affordable mobile devices and cloud computing. In this paper we survey existing mobile cloud computing applications, as well as speculate future generation mobile cloud computing applications. We identified five major challenges, namely, code/computation offloading, task-oriented cloud services, elasticity and scalability, security, and cloud pricing, for building the next generation mobile cloud computing applications. Network bandwidth and computational resource is shared among all the mobile devices, a scheduling scheme are needed to ensure that multiple mobile devices can efficiently offload tasks to local mobile clouds, satisfying the tasks time constraint while keeping low-energy consumption. Two critical challenges need to be solved: (1) estimation of the energy consumption and completion time for tasks to be scheduled, (2) schedule the tasks from multiple source nodes to an appropriate device to accomplish the computation and receive the results. We provide insights for the enabling technologies and challenges that lie ahead for us to move forward from mobile computing to mobile cloud computing for building the next generation mobile cloud applications.***

Key Words: *energy consumption, Mobile Computing, Cloud Services, IAAS, PAAS, SAAS, Task Scheduling, Offloading Elasticity*

1. INTRODUCTION

Mobile devices have become a crucial part of our daily life nowadays. More than 1.99 billion mobile phones and tablets will be sold worldwide in 2013, according to Gartner's analysts. As the capabilities of mobile devices advance (in terms of CPU power, network connectivity and sensors), people increasingly use them for the tasks such as emailing, web surfing, gaming etc. Although there have been many advances in technology, mobile devices will still be resource poor, 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. One solution to overcome these resource limitations is mobile cloud computing, which allows the mobile device to offload the tasks to more powerful resource devices (i.e., servers), as

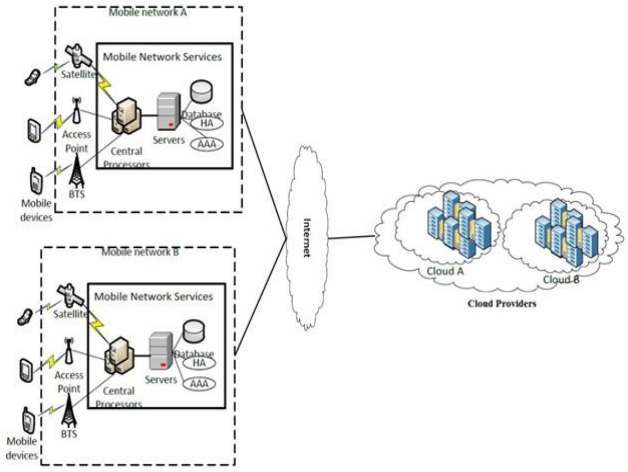

Figure 1, Mobile cloud computing architecture

Fig1: Mobile Cloud Computing Architecture

Augmenting the mobile device's capabilities by using resource providers other than the mobile device itself to host the execution of mobile applications. Since offloading migrates computation to more resourceful computers, it involves making decisions regarding whether and what computation to offload. Therefore, the key challenge is to carry out a cost-benefit analysis to weigh the benefits of offloading against the cost of remote execution with user specific requirement. The decisions are usually made by analyzing parameters including bandwidths, server speeds, available memory, server loads and the amount of data exchanged between servers and mobile systems.

Offloading becomes an attractive solution for meeting stringent response time requirement in mobile systems as applications become increasingly complex. For example, in real-time moving object recognition and tracking system, a robot needs to recognize an object then adjust its speed and direction to track the object's movement. If the robot's processor is too slow, the recognition computation might not be able to be completed before the object moves out of the surveillance range. Much research has been done in mobile cloud to enable the mobile devices to run real-time applications such as object recognition, augmented reality, disaster forecast, and real-time video coding.

Energy consumption is another factor which needs to be taken into consideration when designing mobile cloud systems. Even though battery technology has been steadily improving, it has not been able to keep up with the rapid growth of power consumption of these mobile systems. Energy is still a primary constraint for mobile systems. Therefore, offloading is also used to save energy by migrating the computation tasks with high energy consumption to servers. Many scheduling algorithms have been proposed to make offloading decisions to improve performance or save energy

One type of mobile cloud architecture is to connect the mobile device to the remote server, which is typically far away from the mobile user. A drawback of this architecture is that the high Wide Area Network (WAN) latency makes it unsuitable for real-time applications. Another type of mobile cloud architecture is to consider other mobile devices themselves also as resource providers of the cloud, forming a mobile peer-to-peer network. This network is built based on the Mobile Ad Hoc Network (MANET). An MANET does not rely on pre existing infrastructure, such as routers in wired network. Instead, each mobile node participates in routing by forwarding data for other nodes. To reduce the latency in offloading process in MANETs, it is preferred to offload tasks to nearby devices. Therefore the second type of mobile cloud architecture will be more suitable to support real-time applications, which demand faster responsiveness.

II. RELATED WORK

In this chapter, the existing research on mobile cloud computing will be reviewed, including the following topics: mobile cloud architectures, task-scheduling in mobile cloud system and context awareness in mobile clouds.

2.1 Mobile cloud architectures

There are two typical types of mobile cloud architectures nowadays. The first one is to connect the mobile device to the remote cloud as in Figure 1. The remote cloud is defined as a powerful server or a cluster of computer hardware and software that offer the services to the mobile device users. By leveraging infrastructures such as Amazon's EC2 cloud [14] or Apple iCloud [15], computationally expensive tasks can be offloaded to the cloud so that the capabilities of mobile devices get improved. For example, Apple Siri runs its sophisticated voice recognition feature remotely and then returns the result to the user. There is large amount of research regarding implementing this architecture. However, as mentioned in Chapter 1, this architecture suffers from the long latency. A detailed analysis on why long WAN latencies are a fundamental obstacle in mobile clouds can be found in [19]. Another drawback of this architecture is that it relies on the Internet access. 3G and WLAN are two most popular ways that mobile devices connect to the internet. 3G covers larger area but consumes more energy while WLAN consumes less energy but the transmission range is limited. It is not practical to expect that a good Internet connection is always available. Let alone in battle field, disaster scene and rural area where no Internet access is present but intensive computations are still needed.

- link to the central node is shared among all participating nodes which causes the bottleneck problem;
- Unrealistic deployment of the central node in mobile scenarios. Only the mobile nodes within the transmission range of the central node can join the local mobile cloud, which limits the mobility of the network.

Thus, centralized scheduler is not adequate for the mobile cloud system because of the nature of the dynamic environment in local mobile clouds.

In contrast, decentralized schedulers negate the limitations of centralized schedulers with respect to scalability, autonomy, and most importantly the adequacy for the dynamic local mobile cloud. It does not rely on one single central node. The decentralized scheduler can work as long as a group of mobile nodes are connected, which make the deployment much easier. A decentralized scheduling approach assumes that each participating node is autonomous and has its own controller that derives its scheduling decision based on its policies. However, if the

decisions are taken by several independent nodes, it might be the case that these units aim at optimizing their own objectives rather than the performance of the system as a whole. Such situations call for models and techniques that take the strategic behavior of individual units into account, and simultaneously keep an eye on the global performance of the system. b. Scheduling objective in mobile cloud system

The Eqn. (1) has been used in as a condition for offloading to improve tasks completion time:

$$w \quad d; \quad w$$

Suppose w is the amount of computation and d_t is the size of data needs to be transmitted for remote execution. Let s_m be the speed of the mobile device and s_s be the speed of the server. B is bandwidth between the mobile device and the server. The left side is the time needed to execute the task on the mobile device itself. The right side is the time needed to execute the task on the server plus the transmission time.

Suppose the task's computation requires C instructions. Let S and M be the execution speed in instructions per second, of the cloud server and the mobile device, respectively. The server and mobile device exchange D bytes of data and the network bandwidth is B bps. The mobile device consumes, in watts, P_c for computing, P_t while being idle, and P_{tr} for sending and receiving data. When the server performs the computation, the amount of energy saved is given by Different scheduling schemes with more complex criteria are proposed in several other papers. A customizable task scheduler is proposed in [11] using Map Reduce framework for local mobile clouds. It is customizable because the user can optimize multiple objectives such as power consumption and (or) throughput through adjusting the corresponding coefficients in the objective function in Eqn. (3), where a_1,a_2,a_3,a_4,a_5 are variable coefficients, F_1 is a function of the task processing time on node j, F_2 is a function of task queuing time on node j, F_3 is a function of the communication cost of sending a task from node i to node j, F_4 is a function of battery level on node j, F_5 is the energy consumption of sending a task from node i to node j.

$$F(i) = Min[a_1 xF_1\{PT_j) + a_2 xF_2\{Qj) + a_3 xF_3(Comm[i])) + a_4 xF_4(Ej, Cj) + a_s xF_s(EComm[ij])] \quad for\ i,j = 1,2 \ldots n, ji \qquad (3)$$

Symbol	Description
V	Set of all wireless nodes in the local mobile cloud
E	Set of all wireless links in the local mobile cloud
$G(V,E)$	Undirected topology graph that is composed of node set V and edge set E
$mips_u$	Average processing speed of node u in terms of millions of instructions per second (mips)
e_u	Average energy consumption per million instructions of node u
r_t	Radio transmission range of node u
$B_{u,v}$	Bandwidth (in bps) between node u and node v
e_t	Average energy consumption of node u to transmit one byte
e_r	Average energy consumption of node u to receive one byte
$t_{q,u}$	Queuing time of node u
J	Set of tasks arriving at V
D_j	Data size of task j
C_j	Computation amount of task j in number of instructions
T_j	Time constraint set for task j
t_{margin}	Time margin used when comparing the estimated task completion time with T_j
P	Set of all processing nodes in the local mobile cloud, $P \subseteq V$
S	Set of all source nodes in the local mobile cloud, $S \subseteq V$

Table 1: NOTATION

III. PREVIOUS IMPLEMENTATION

Cloud computing has been recognized as a prospective computing paradigm because of its potential benefits such as on-demand service, ubiquitous network access, location independent resource pooling, and transference of risk. In the cloud computing paradigm, there are a service provider who owns and manages the centralized computing and storage resources in the cloud, and users who have access to those resources over the Internet. With the development of wireless communication technology such as 3G, Wi-Fi, and 4G, the mobile cloud computing (MCC) paradigm has emerged to shift the processing, memory, and storage requirements from the resource-limited mobile devices to the resource-unlimited cloud computing system.

Existing MCC Applications We have witnessed a number of MCC applications in recent years, including mobile commerce, multimedia sharing, mobile learning, mobile sensing, mobile healthcare, mobile gaming, mobile social networking, location-based mobile service, and augmented reality. Mobile commerce, such as e-banking, e-advertising and e-shopping, uses scalable processing power and security measures to accommodate a high volume of traffic due to simultaneous user access and data transaction processing.

Multimedia sharing provides secure viewing and sharing of multimedia information stored on smart phones while providing administrative controls to manage user privileges and access rights necessary to ensure security. Mobile learning allows a thin terminal to access learning materials on the cloud any time and any place. Mobile sensing utilizing sensor-equipped smart phones to collect data will revolutionize many MCC applications including healthcare, social networking, and environment/health monitoring. Mobile healthcare allows an enormous amount of patient data to be stored on the cloud

instantaneously. A doctor can conveniently look at the patient records on his/her mobile device for remote diagnosis or monitor a patient's status for preventive actions. Mobile gaming achieves scalability by leveraging scalable computation and instantaneous data update on the cloud side and screen refresh at the mobile device side. Mobile social networking allows a group of mobile users to upload audio/video/multimedia data for real-time sharing, with cloud computing providing not only storage for data, but also security to protect secrecy and integrity of data.

3.1 Task-Oriented Mobile Services

The second design issue is to provide mobile devices flexible task-oriented mobile services for offloading applications to the cloud. Mobile devices have severe limitations in computation, memory and display capability. The wireless network environment often is changeable with unreliable communication service. Mobile users also tend to use smart phones to do certain tasks but not all tasks. Hence, mobile devices require task-oriented mobile services.

While cloud computing offers SaaS, PaaS, and IaaS services, MCC must tailor task-oriented mobile services specifically to the need of mobile users. Below we survey existing work in the literature on Mobile-Data-asa-Service, Mobile-Computing-as-a-Service, Mobile-Multimedia-as-a-Service and Location-Based Mobile Cloud Services. We note that these task-oriented mobile services do not cover the entire spectrum and more research effort is called for to create more human centric task-oriented mobile services, thus expanding the list of task-oriented mobile services used by mobile users.

3.2 Network Model

Definition 1.A MANET that consists of a number of wireless nodes can be modelled by an undirected communication graph *G(V, E).* Given a node $u \; E \; V$ and a node v E F , we have (it,$v) \; E \; E$, if and only if $dis(u.v) < r_t$, where $dis(u.v)$ is the Euclidean distance between node u and node v . That is, to establish a direct communication between any two nodes, the distance between them has to be within their radio range r_t. The computational ability of the processing node u is denoted by $mips_u$ (Million instructions per second). According to [33], Eqn. (4) gives that the power e consumed by a CMOS processor, in watts, is equal to the activity factor a of the system (percentage of gates that switch for each cycle, on average 50%) multiplied by the capacitance C of the CPU, the voltage ^squared, the frequence

3.3 Statement

With the network model and task model, the energy consumption and completion time can be calculated. The energy consumption of executing a task includes two parts:

1) Computation energy, which is the energy dissipated for executing the task by the processing node *u;*

$$Computation\ Energy\ j = e_u \; x \; C\ j$$

2) Communication energy, which is the energy consumed in communication for the offloading process. The communication energy dissipated is proportional to the amount 18 of data transmitted or received. $H(j)$ is the hop count from the source node to the processing node.

$$Communication\ Energyj = 2 \; x \; H(j) \; x \; (e_t + e_r) \; x \; D\ j$$

The total energy *Task Energy j* consumed by task *j* is the

Summation of these two types of energy:

$$Task\ Energy\ j = Computation\ Energy\ j + Communication\ Energy\ j$$

IV. PROPOSED WORK

The application completion time constraint is a hard constraint, and therefore it should be addressed in the first place. Offloading computation intensive tasks onto the cloud may result in less application completion time. However, the offloading decision should be made judiciously considering the delay due to uploading/ downloading data to/from the cloud.

The total energy consumption in a mobile device for executing an application, including the energy consumed both by the processing units (i.e., the potentially heterogeneous cores in the mobile device) and by the RF components for offloading tasks, is the objective function to be minimized. From the perspective of energy consumption, offloading tasks onto the cloud saves the computation energy but induces the communication energy.

To solve the above task scheduling problem, we propose a distributive adaptive probabilistic scheduler which consists of two phases, namely the resource discovery phase and the adaptive probabilistic scheduling phase. In the resource discovery phase, source nodes are able to get the context information about the nearby processing nodes. In the adaptive probabilistic scheduling phase, the scheduler will choose one processing node to execute the task j. These two phases work together to ensure the performance in local mobile clouds gets improved.

4.1 A. Phase I: Resource discovery phase

The proposed resource discovery scheme is based on QoS OLSR. There are two kinds of control messages carrying resource information:

• Modified Hello Messages, which are sent locally (i.e. broadcasted to one-hop neighbors) to enable a node to discover its local neighborhood (as HELLO messages in the QoS OLSR protocol);

• Modified Topology Control (TC) Message, which are sent to the entire network through Multipoint Relay (MPR) nodes [32] to allow the distribution of the topology and context information to all the nodes (as TC messages in the QoS OLSR protocol.

4.2 Resource Discovery Algorithm

Input: **Control messages (Hello message or TC message)**

Output: **Neighbor table(The table storing the node parameter for each node)**

Function: **Handle control messages and update the neighbor table at node u**

Procedure body:

```
{
        initialize the neighbor table
        listen control messages
        if (Message Type== HELLO_MESSAGE)
        {
         update the neighbor table
         if (node u is an MPR node)
         construct/update TC message
                }
        if (Message Type == TC_MESSAGE)
{
        update neighbor table
         if (node u is an MPR node)
         forward the TC message to all of its neighbors
                }
```

} Our proposal is toward an extension of the routing table by adding the following parameters of each neighbor node into the two types of control messages:

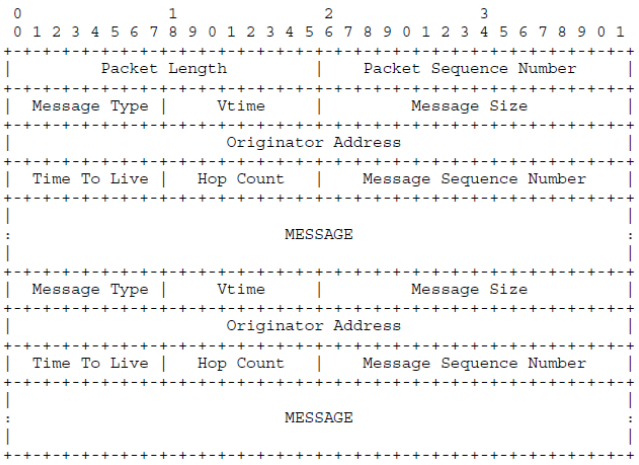

Fig2: MESSAGE FORMAT

The description of each field can be found in [35]. The 'MESSAGE' field carries the control message. The difference between original control messages and modified control messages are given below:

- **Device parameter:** indicate the processing speed $mips_u$ and energy consumption coefficient e_u of node it.

- **Queue length:** current queue time t_{qu} at node it. The basic layout of any packet in QoS OLSR is as follows (omitting IP and UDP headers):

Original Hello Message:

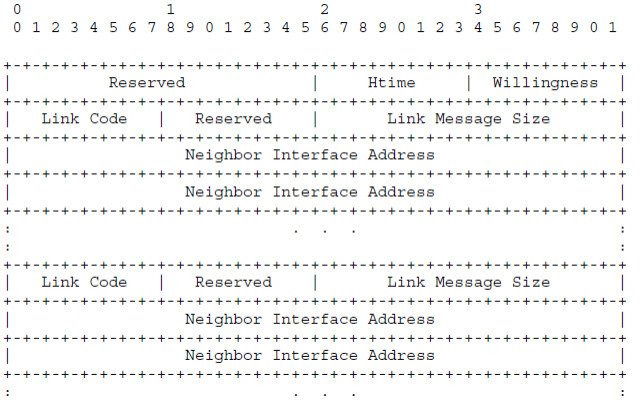

Fig3: Time to Live

This is sent as the data-portion of the general packet format with the "Message Type" set to HELLOMESSAGE and the Time to Live (TTL) field set to 1.

Modified Hello Message:

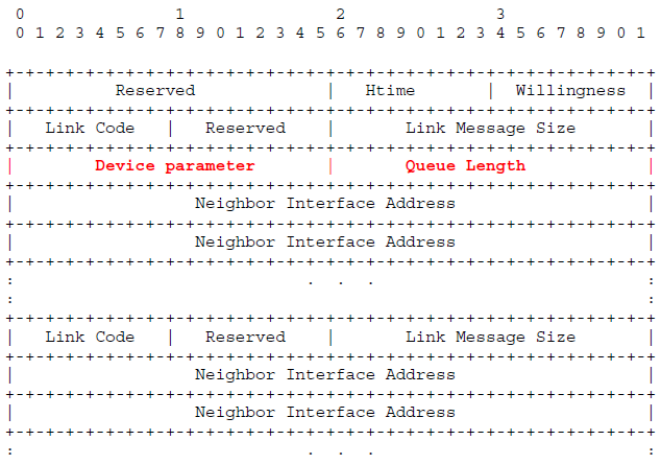

Fig 4: Modified Hello Message

By adding device type and queue length, the current context information of the node can be updated at its neighboring nodes.

Modified Topology Control (TC) Message, which are sent to the entire network through Multipoint Relay (MPR) nodes [32] to allow the distribution of the topology and context information to all the nodes (as TC messages in the QoS OLSR protocol.

Original Topology Control (TC) Message:

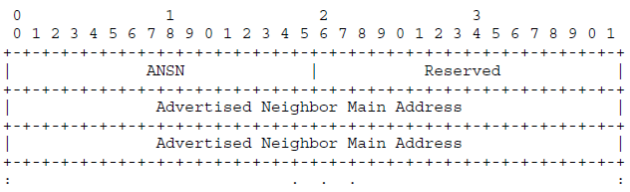

Fig 5: Original Topology Control

4. After executing the algorithm listed in Figure 4, all nodes in the network shall obtain a neighbour table in its local memory.

Parameter	Value
Task arrival intervals	0.5s
Task data size	1000B
Task computation amount	200 million instructions(MI)
Task time constraint	1s
$e_{n0}/e_{n1}/e_{n2}/e_{n3}$	0.25/0.16/0.09/0.04 J/MI
$mips_{n0}/mips_{n1}/mips_{n2}/mips_{n3}$	500/400/300/200 MI/s

Table2: Parameter settings of the illustrative example

V. EVALUATION RESULT

Simulated local mobile cloud contains a group of nodes, each of which is capable of transmitting radio signals up to approximately 40 meters over an 11Mbps 802.11g wireless channel. Two network scenarios (with different node densities) are simulated. The first scenario, referred to as the small network, is created by randomly placing 10 nodes in a 200m x 200m area. Among these 10 nodes, 2 nodes are source nodes and 8 nodes are processing nodes. The second scenario, referred to as the large network, is created by randomly placing 20 nodes in a 200m x 200m area. Among these 20 nodes, 4 nodes are source nodes and 16 nodes are processing nodes.

There are two mobility patterns: the first one is the stationary network in which all nodes are stationary. The second one is the mobile network in which every mobile node $u \; E \; \boldsymbol{V}$ moves within the area according to the following pattern. It moves along a straight line for a certain period of time before it makes a turn. This moving period is a random number, normally distributed with average of 5 seconds and standard deviation of 0.1 second. When it makes a turn, the new direction (angle) in which it will move is a normally distributed random number with average equal to the previous direction and standard deviation of 30 degrees. Its speed is also a normally distributed random number ranging from 1 to 3 (m/s). A new such random number is picked as its speed when it makes a turn. All nodes can move to a random direction within the area with a speed uniformly distributed between 1 m/s and 3 m/s. All nodes have the same initial battery level. The energy of a node is only consumed when the node processes a task or when the node transmits/receives a message, which can be calculated using Eqn. (6)-(8). The energy consumed in moving is ignored here.

Parameter	Value
Topology	Random
Network area	200m*200m
Network size	Small: 10 nodes /Large: 20 nodes
Communication range	Approximately 40 meters
Task data size	Varying from 1000 B to 8000 B
Task computation amount	Varying from 50 MI to 350 MI
Task time constraint	0.5s
Task arrival interval	Exponential distribution X =0.2s
Task arrival duration	200s
Computation ability of node u $\boldsymbol{mips_u}$	Normal distribution in (1000,300) mips
Computation energy per MI of node	$10^{-8} x \boldsymbol{mipSy^2}$ J
Maximum bandwidth	11Mbps

Table 3: Parameter settings in simulation

5.1 Effect of varying Topology Control message interval

As introduced in Chapter 3, we modified the QoS OLSR by adding extra bytes to control messages. The influence of the modification is going to be evaluated in this section. In the original QoS OSLR, by default the Hello message interval is 2s and TC message interval is 5s.In the modified QoS OLSR scheme, the Hello message's interval is half of the TC message interval. The reason is that TC message is sent to the whole network with neighbor information updated at least once. Considering the tight time constraints (0.5s), we expect the control messages to be more frequently sent so the context information can be updated on time.

In order to gauge the overhead, we measure the number of bytes in the control messages, including HELLO messages and TC messages, in a small network. The *average overhead traffic* is defined as the number of bytes in the control messages transmitted per second in the whole network. The average overhead traffic shows how much network bandwidth is used for overhead messages. Figure 8 plots the average overhead traffic of the original QoS OLSR and the proposed modified QoS OLSR vs. varying TC message interval.

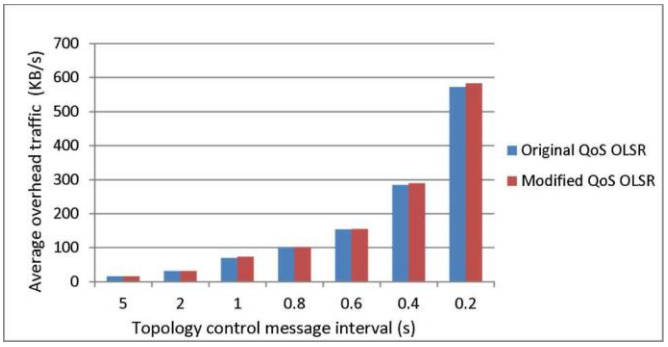

Fig6: AVERAGE ENERGY PER SUCCESSFUL

The round robin scheduler does not rely on the control messages. It sends tasks to nearby nodes one after another. Thus its task completion rate remains the same. The proposed adaptive probabilistic scheduler has the highest completion rate for most cases. It has 16.6% and 3.6% higher completion rate than the greedy scheduler and the probabilistic scheduler, respectively. The reason is that with an increased t_{margin}, the scheduler will send tasks to a more computationally powerful node.

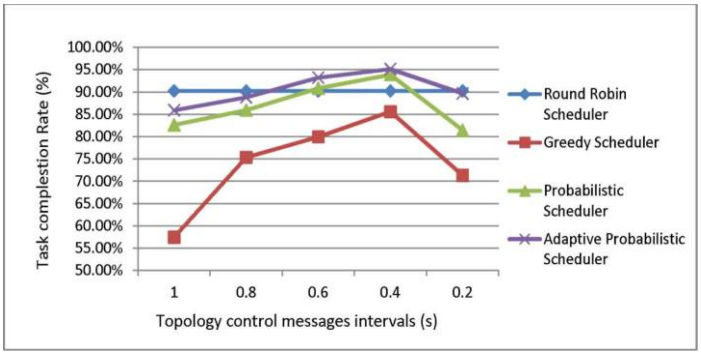

Fig7: AVERAGE ENERGY PER SUCCESSFUL

shows that the average energy per successful task result is correlated with the task completion rate result. Except for round robin scheduler, the average energy per task of the other three scheduler's first decreases as the TC message interval is reduced but increases significantly when the TC message interval reaches 0.2s. Among all four schedulers, the average energy per task of the adaptive probabilistic scheduler is the best. When the TC message interval is 0.4s, the adaptive probabilistic scheduler has the highest completion rate 95.14% and the lowest energy per successful task 1.60 J.

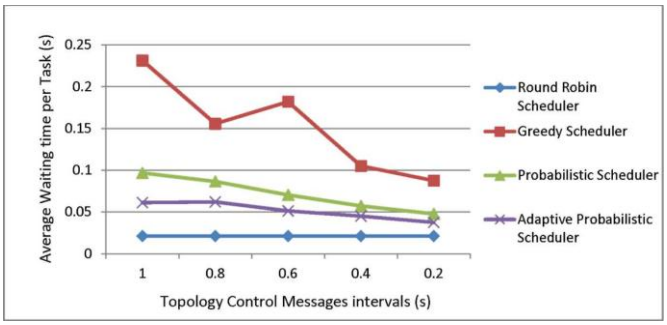

Fig8: AVERAGE WAITING TIME PER TASK VS. TC MESSAGE INTERVAL

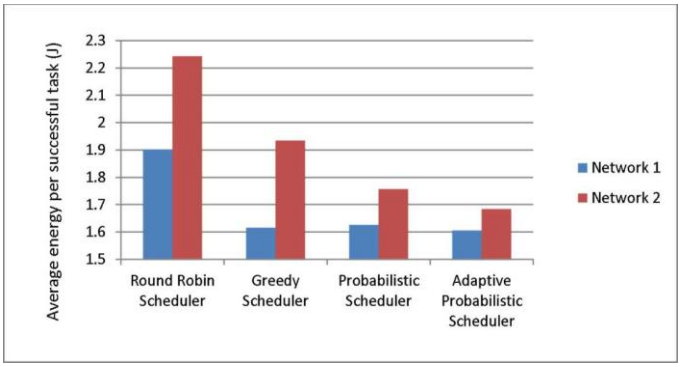

Fig8: AVERAGE ENERGY PER SUCCESSFUL TASK IN SMALL AND LARGE NETWORK

CONCLUSION

In this thesis, we propose an adaptive probabilistic task scheduler for real-time applications in local mobile clouds. Based on QoS OLSR, source nodes receive periodical control messages to discover and update nearby resource information. The scheduler first estimates the task completion time and energy consumption at each potential processing node based on the context information collected through the modified QoS OLSR. Next, it schedules the current task to the proper processing node in a probabilistic way. Then it adaptively adjusts its time margin parameter to improve performance under the unpredictable network conditions.

Overall, the observed experimental results confirm that the adaptive probabilistic scheduler is able to reduce the average energy per successful task while maintain a high task completion rate. Furthermore, the performance benefit is more significant when the number of source nodes increases. In addition, the proposed scheduler can adjust itself to work for both stationary and mobile network scenarios. It also shows high adaptability with different task types. The adaptive probabilistic scheduler is a promising approach for real-time applications due to its scalability and flexibility in local mobile cloud.

REFERENCES

[1] Gartner: the world's leading information technology research and advisory company. [Online]. http://www.gartner.com/

[2] Eduardo Cuervo et al., "MAUI: making smartphones last longer with code offload," in *Proceedings of the 8th international conference on Mobile systems, applications, and services,* 2010, pp. 49-62.

[3] Yamini Nimmagadda, Karthik Kumar, Yung-Hsiang Lu, and C. S. George Lee, "Real-time Moving Object Recognition and Tracking Using Computation," in *The 2010 IEEE/RSJ International Conference on Intelligent Robots and Systems,* 2010, pp. 2449-2455.

[4] Roelof Kemp et al., "EyeDentify: Multimedia Cyber Foraging from a Smartphone," in *ISM '09 Proceedings of the 2009 11th IEEE International Symposium on Multimedia,* 2009, pp. 392399.

[5] Tim Verbelen, Pieter Simoens, De Turck Filip, and Dhoedt Bart, "Cloudlets: Bringing the Cloud to the Mobile User," in *Proceedings of the third ACM workshop on Mobile cloud computing and services,* 2012, pp. 29-36.

[6] Romano Fantacci, Daniele Tarchi, and Andrea Tassi, "A novel routing algorithm for mobile pervasive computing," in *Global Telecommunication Conferrence,* 2010, pp. 1-5.

[7] Shu Shi, Cheng-Hsin Hsu, Klara Nahrstedt, and Roy Campbell, "Using graphics rendering contexts to enhance the real-time video coding for mobile cloud gaming," in *MM '11 Proceedings of the 19th ACM international conference on Multimedia,* 2011, pp. 103-112.

[8] Karthik Kumar, Jibang Liu, Yung-Hsiang Lu, and Bharat Bhargava, "A Survey of Computation Offloading for Mobile Systems," in *Mobile Networks and Applications,* 2013, pp. 129-140.

[9] Asaf Cidon, Tomer M London, Sachin Katti, Christos Kozyrakis, and Mendel Rosenblum, "MARS: adaptive remote execution for multi-threaded mobile devices," in *MobiHeld '11 Proceedings of the 3rd ACM SOSP Workshop on Networking, Systems, and Applications on Mobile Handhelds, Article No. 1.*

[10] E.E. Marinelli, "Hyrax: cloud computing on mobile devices using MapReduce," in *Masters Thesis, Carnegie Mellon University,* 2009.

[11] Suraj Sindia et al., "MobSched: Customizable Scheduler for Mobile Cloud Computing," in *45th Southeastern Symposium on System Theory,* 2013, pp. 129-134.

[12] Xiaoshuang Lu, Hossam Hassanein, and Selim Akl, "Energy aware dynamic task allocation in mobile ad hoc networks," in *International Conference on Wireless Networks Communications and Mobile Computing,* 2005, pp. 534-539.