International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395-0056
Volume: 06 Issue: 10 | Oct 2019 www.irjet.net p-ISSN: 2395-0072

Collaborative Task Execution for Application as a General Topology in Mobile Computing under Communication Cost Model

J. Manikandan[1], A. Srinish Reddy[2], Goddeti Mallikarjun[3]

[1,2,3]*Assistant Professor, St. Martins Engineering College, Dept. of CSE, Secunderabad.*

Abstract - This paper investigates collaborative task execution between a mobile device and a cloud clone for mobile applications under a stochastic wireless channel. *Mobile cloud computing has been proposed as an effective solution to augment the capabilities of resource poor mobile devices. We aim to minimize the energy consumption on the mobile device while meeting a time deadline, by strategically offloading tasks to the cloud. We aim to find out the execution decision for each task to minimize the energy consumption on the mobile device while meeting a delay deadline. We formulate the collaborative task execution as a delay-constrained workflow scheduling problem. We leverage the partial critical path analysis for the workflow scheduling; for each path, we schedule the tasks using two algorithms based on different cases. For the special case without execution restriction, we adopt one-climb policy to obtain the solution. For the general case where there are some tasks that must be executed either on the mobile device or on the cloud, we adopt Lagrange Relaxation based Aggregated Cost (LARAC) algorithm to obtain the solution. We show by simulation that the collaborative task execution is more energy-efficient than local execution and remote execution.*

Key Words: Collaborative task execution, Mobile cloud computing, energy efficiency, general topology, task execution.

1. INTRODUCTION

Mobile devices, owing to the latest technology advances in wireless communication and computer architecture, are being transformed into a ubiquitous computing platform. Cisco VNI report [1] predicts that the number of mobile users will continue to increase in the following years, with 3.8 billion in 2012 growing to 4.6 billion by 2017. The proliferation of mobile devices has driven the emergence of a large number of mobile applications. Cisco VNI report [1] predicts that by 2019 there will be nearly 1.5 mobile devices per capita. Indeed, according to the statistics of Apple and AppBrain, there were 1.3 million applications on iPhone App Store by July 2014 [2] and more than 1.5 million Android applications on Google Play by April 2015 [3]. These numbers could continue to grow, resulting in great impact on ubiquitous computing.

However, mobile devices, due to the small physical size, are resource constrained for the applications that are computation intensive. Although there has been technical innovation on mobile devices (e.g., smartphones and tablets), computation capability and memory capacity still fall behind their desktop counterparts. Furthermore, as the battery system has not been improved on pace with the computation capability and memory capacity, the short battery lifetime has become a bottleneck of running computation intensive mobile applications [4]. The mobile applications, such as image processing and object recognition, can consume a large amount of energy on mobile devices. Therefore, energy issue of mobile devices should be taken into consideration for designing mobile applications. Mobile cloud computing [5], [6] has been touted as an effective solution to mitigate the resource constraints of mobile devices. Two categories of cloud-assisted mobile application platforms have been proposed, including infrastructure-based cloud and ad-hoc virtual cloud. Empowered by the remote servers, the infrastructure-based cloud provides sufficient computation resources to mobile devices, such as MAUI [7], Clone Cloud [8], Cloudlets [9] and Weblet [10]. Rather than relying on remote servers, the ad-hoc virtual cloud, is formed by a group of mobile devices nearby that work cooperatively to accomplish application offloading [11], [12]. By application offloading, the tussle between the computation intensive mobile applications and resource poor mobile devices can be alleviated. Nevertheless, it is not always energy efficient to offload applications to the cloud for execution [4]. The decision of application offloading, i.e., offloading policy, should be investigated to consider the tradeoff between the amount of computation and communication.

2. RELATED WORK

Some researchers have provided analysis of offloading decision for coarse-grained applications. Kumar et al. in [4] presented an energy model to analyze whether to offload applications to the cloud. The analysis was made based on the tradeoff between computation energy for mobile execution and communication energy under a static network for cloud execution. Miettinen et al. in [21] had similar analysis and demonstrated that workload, data communication patterns and technologies are the main factors that affect the energy consumption of mobile applications for application offloading. But its analysis was roughly based on measurements and investigations. In [13], the offloading decision was made by solving two optimization problems, i.e., optimal clock frequency configuration on the mobile device for local execution and optimal data transmission over stochastic wireless channel for remote execution. The policy was then to

select the execution mode that results in less energy consumption on the mobile device. However, it can only be applied for the application as a single task. For fine-grained applications, graph representation and integer program formulation have been typical approaches for obtaining the offloading policy.

Graph representation: Given an application, Li et al. in [22] constructed a cost graph and divided the application into server tasks and client tasks for execution. Branch-and bound algorithm was used to solve the optimization problem for minimizing the energy consumption of computation and data communication cost for task execution. In [23], Wang and Li modeled a task graph and proposed a min-cut approach to minimize the cost for task execution between the mobile device and the cloud. Giurgiu et al. in [16] constructed a graph to represent an application and presented algorithms to find a cut in the graph for the application execution between the mobile device and the cloud. The cut was determined by solving an unconstrained optimization problem, the objective function of which is a combination of the interaction time and the amount of exchanged data.

Nevertheless, their cut-based approach cannot be used in our research problem, since they do not consider the energy consumption and time delay constraint.

3. SYSTEM MODELS AND PROBLEM FORMULATION

In this section, we present the models, including task graph model, computation cost model and communication cost model, and problem formulation for the collaborative task execution between the mobile device and the cloud clone.

3.1 Task Graph Model

A mobile application can consist of a set of tasks in the granularity of either method [7] or module [16]. Within an application, a task can call other tasks for execution. The invoke of a task needs the output data from other tasks. Thus, the tasks are connected through the corresponding dependencies (shown in Figure 1 (a) and Figure 1 (b)).

Fig. 1: Tasks within the application. A node represents a task and an arc denotes the data dependency between tasks. (a) The completion of a task can call other tasks for execution. (b) The execution of a task can require the output data of other tasks.

To model the dependencies among the tasks, we represent the tasks by a directed acyclic graph *G = (V,A)*. We have the following assumptions for the task graph model:

- We do not consider the cyclic graph, where methods or modules can be invoked recursively; indeed, we can regard them as single tasks.
- We assume that the tasks have been identified based on the design and implementation of the mobile application; we do not consider how we partition the application into tasks, but focus on the scheduling for the tasks.
- We assume that we have the information in advance, for which tasks must be executed on the mobile device and on the cloud clone, respectively.

Without loss of generality, we suppose that there are n tasks in the application. We denote V as the node set for the tasks, i.e., $V = \{v_i\}$. In this paper, we use the terms node and task interchangeably. We add two dummy nodes into the graph, i.e., v0 and vn+1 to denote the initialization and termination of the application, as shown in Figure 2. We denote A as the arc set for data dependencies among the tasks. An arc a_{ij} in A indicates the dependency between the adjacent task I and task j, which requires that task j cannot be started if its parent, task i, is not completed. In addition, we denote X as the execution decision for the tasks, i.e., $X = \{x_i\}$. The value of xi is either 0 denoting that task i is executed on the mobile device, or 1 denoting that on the cloud clone. As the initialization and the termination of the mobile application should be on the mobile device, we have $x_0 = 0$and $x_{n+1} = 0$ for the two dummy tasks.

The task graph model has been demonstrated in the interactive mobile applications of object, pose and gesture recognition [15], [26]. For example, in object and pose recognition [26], an image goes through a scale transformer before a set of SIFT features are extracted from the image (corresponding to Figure 1(a)); and these SIFT features will be delivered to the component of feature merger for finding objects of interest (corresponding to Figure 1(b)). After that, the features of each object are clustered to separate the objects and a consensus algorithm is subsequently used to recognize each object and determine its pose. As such, the process of object and pose recognition, consisting of image scale transformer, feature extractor, feature merger, cluster and recognition algorithm, can be modeled as a general task topology. Those two works, however, only aim to achieve the maximum throughput of the application, without considering the energy consumption on the mobile device. In this paper, we aim to reduce the energy consumption of completing the tasks within the application by collaborative task execution.

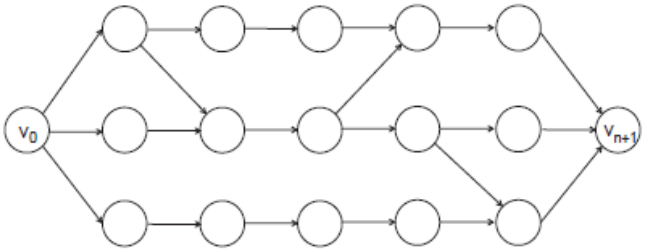

Fig. 2: A graph representation of tasks within an application as a general topology.

3.2 Communication Cost Model

Communication cost of tasks depends on their execution decisions. We assume that the communication cost is negligible if the two adjacent tasks are assigned to the same device. In other words, for the tasks v_i and v_j with (v_i, v_j) 2 A, if $x_i = x_j$, then the data communication time between task i and task j is zero and thus, the resulted communication energy is also zero. We consider the communication time and communication energy on the mobile device for the task execution as follows.

First, we denote $T^{comm}_{v_i,v_j}(x_i, x_j)$ as the communication time on the mobile device for data communication between tasks i and task j, where v_j is a child node of v_i. Then, the communication time between task i and task j is given by,

$$T^{comm}_{v_i,v_j}(x_i, x_j) = \begin{cases} 0, & x_i = 0, x_j = 0 \\ \beta_{ij} R^{-1}, & x_i = 0, x_j = 1 \\ \beta_{ij} R^{-1}, & x_i = 1, x_j = 0 \end{cases} --- (1)$$

Where β_{ij} is the output data size of task i to task j and R is the average rate of data transmission. We assume that we have the information of the average data rate over the wireless channel by profiling and then the communication time can be approximated by Eq. (1).

3.3 Problem Formulation for Collaborative Task Execution

In this paper, we investigate how to minimize the energy consumption on the mobile device while completing all the tasks within the delay deadline, by determining the execution of each task within the application. Specifically, the energy consumption of the application execution is the sum of the weights of nodes and arcs in the graph G. The completion time of the application execution is when the last task in the graph (i.e., v_{n+1} in Figure 2) completes. The collaborative task execution can be formulated as a constrained workflow scheduling problem. Mathematically, we have

$$T(X) = T^{finish}_{v_{n+1}} \leq T_d --- (2)$$

$$x_i = 0; \forall v_i \in M --- (3)$$

$$x_j = 1; \forall v_j \in C --- (4)$$

Where $T^{finish}_{v_{n+1}}$ the finish time of task v_{n+1}, M is denotes the set for the tasks that must be executed on the mobile device, and C denotes the set for the tasks that must be executed on the cloud clone. We aim to find a schedule X that minimizes the energy consumption on the mobile device while meeting the delay deadline (Eq. (2)).

Notice that some tasks may need to access the sensor on the mobile device or access the database in the cloud; thus, we include Eq. (3) and Eq. (4) as the constraints into the optimization problem, respectively. Specifically, Eq. (3) indicates that some tasks must be executed on the mobile device if M≠ ∅. Similarly, Eq. (4) indicates that some tasks must be executed on the cloud clone if C≠ ∅. We refer Eq. (3) and Eq. (4) as to execution restriction. The optimization problem with the execution restriction is a general case for the previous work [25] as a special case. Since the scheduling problem is NP-complete, we leverage heuristic algorithms.

4. DESIGN OF WORKFLOW SCHEDULING ALGORITHM

In this section, we adopt a two-step approach to design the workflow scheduling algorithm for the collaborative task execution. First, we leverage partial critical path analysis to find the path that has the largest finish time. Second, we schedule the tasks on the partial critical path to achieve the minimum energy consumption. A summary of the design of the scheduling algorithm is given as follows:

- **Partial Critical Path**: Partial Critical Path (PCP) in [19] was proposed as a heuristic algorithm to solve the scientific workflow scheduling problem in the cloud. A partial critical path of a task v consists of critical parent v_0 of the task v and the partial critical path of v_0. In [19], the proposed approach using PCP did not consider the communication cost in the internal cloud and thus data transfer cost between clouds instances is zero. In contrast, for the collaborative task execution, the energy cost between the mobile device and the cloud cannot be ignored.
- **Task Scheduling Strategy**: After finding a partial critical path, we schedule the tasks on the partial critical path for the collaborative task execution. We leverage one-climb policy [14] or LARAC algorithm [20] to obtain the execution decision for each task within the application. If there is no execution restriction (i.e., M =ϕ; and C =ϕ ;) in the optimization problem (we refer it as to special case), we adopt the one climb policy for task scheduling, which indicates that there exists at most one migration from the mobile device to the cloud if ever. However, if there is execution restriction (we refer it as to general case), the one-climb policy may not be applicable and we adopt the LARAC algorithm to schedule the tasks for execution. More details will be elaborated in the following subsections.

4.1 Scheduling Metrics

We define two metrics for the partial critical path analysis. We denote T^{es} for each task as the earliest time for which it will start the execution, and T^{lf} as the latest time for which it will finish the execution. We approximate the earliest start time T^{es} and the latest finish time T^{lf} as follows. Given different execution decisions on the tasks, computation time of the task execution (mobile device or cloud clone) and communication time between tasks are different. Thus, we cannot have the exact T^{es} and T^{lf} . In this case, we compute the T^{es} and T^{lf} by approximation for each unscheduled task [27]. For the approximation, suppose that all the tasks are offloaded to the cloud for execution, except the tasks that must be executed on the mobile device. Then, the earliest start time of each task is

$$T_{v_o}^{es} = 0 - - - -(5)$$

$$T_{v_i}^{es} = \max_{v_j \in P(v_i)} \{T_{v_j}^{es} + T_{v_j}^{comp} + T_{v_i}^{comm}(x_i, x_j)\} - - - (6)$$

Where $C(v_i)$ is the set of child tasks of v_i. The calculation of T^{es} and T^{lf} can help to find out the partial critical path before we can schedule the tasks and the sub-deadline for the task scheduling. In addition, T^{es} and T^{lf} will be updated after the task scheduling.

5. PERFORMANCE EVALUATION

In this section, we present simulation results of the collaborative task execution for the special case and the general case.

5.1 Application Profile

Table 1 illustrates the parameters of the mobile device and the cloud clone in system models mentioned in Section 3. As an example, we consider the application that consists of 11 tasks. Figure 3 shows three task topologies,

- Mesh: a set of linear chains;
- Tree: a tree-based structure;
- General: a combination of the mesh and the tree.

TABLE 1

Parameters of simulation

Data transmission power of the mobile device	$p_s = 0.1W$
Data receiving power of the mobile device	$p_r = 0.05W$
Computation power of the mobile device	$p_m = 0.5W$
Idle power of the mobile device	$p_0 = 0.001W$
CPU frequency of the mobile device	$f_m = 500MHz$
CPU frequency of the cloud clone	$f_c = 3000MHz$

We add two dummy nodes v0 and v12 as task 0 and task12, respectively. The workloads for these two nodes are 0. In addition, the value of the arc between the nodes denotes the amount of data in kb units for transmission if the two corresponding tasks are executed on different devices. Based on these task topologies, we investigate three cases under different execution restrictions as follows:

- Case 1: no execution restriction;
- Case 2: some tasks must be executed on the mobile device;
- Case 3: some tasks must be executed on the mobile device while some are on the cloud clone.

For case 1, we will leverage one-climb policy for the task execution. For case 2 and 3, we will leverage LARAC algorithm for the task execution. As an example, for case 2, suppose M = {v_5, v_{11}}, then we set $x_5 = 0$ and $x_{11} = 0$ to denote that task 5 and task 11 must be executed on the mobile device; for case 3, suppose M = {v_5} and C = {v_{11}}, then we set x5 = 0 and x11 = 1 to denote that task 5 must be executed on the mobile device and task 11 must be executed on the cloud clone. Our approach is not limited to these specific cases.

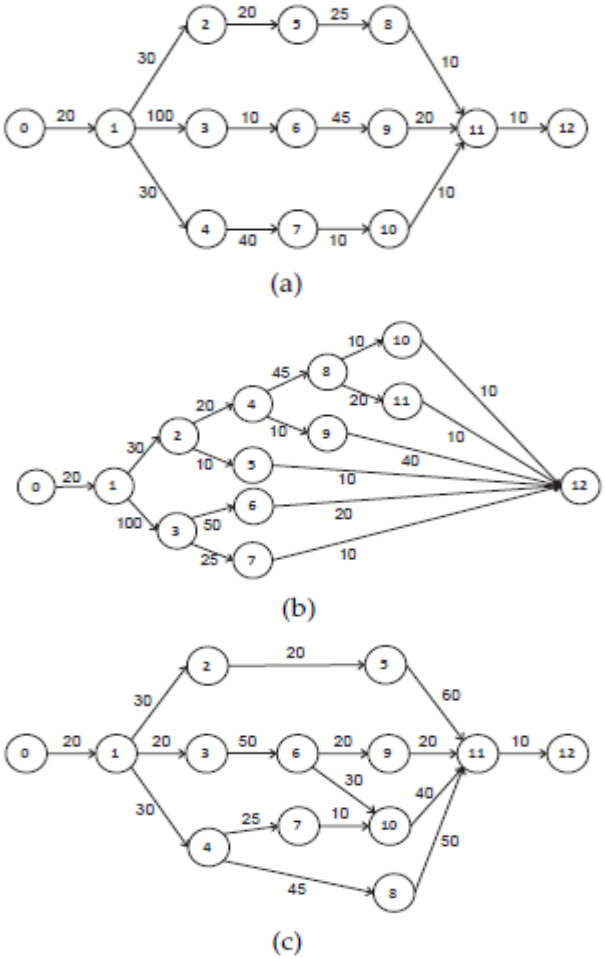

Fig. 3: Task topologies: (a) mesh; (b) tree; (c) general topology. There are 13 nodes in each task topology. v_0 and v_{12} are added into the task topologies as task 0 and task 12, respectively. The workload for each task is ! = [0, 10, 25, 2, 12, 15, 67, 54, 24, 50, 9, 20, 0] M cycles. The value of the arc between the adjacent nodes refers to the data communication.

5.2 Simulation Results

We present the solution of the proposed algorithm in comparison with of the optimal solution; analyze the effect of the execution restriction on the execution decision, and compare the collaborative task execution with the local execution and the remote execution.

5.2.1 Comparison with the Optimal Solution

We compare the solution of the proposed workflow scheduling algorithm with the optimal solution using brute-force enumeration for different cases.

(1) For case 1 under the one climb policy, we observe that with the low data rate (e.g., R = 10kb/s) and high data rate (e.g., R = 100kb/s), our proposed algorithm achieves the same energy consumption as the brute-force enumeration method for the topologies shown in Figure 3. We thus only plot the energy ratio between the proposed workflow algorithm and the brute force enumeration with R = 50kb/s. Figure 4(a) shows that the ratio is 1 for the general topology, indicating that the proposed workflow scheduling algorithm achieves the optimal solution. However, the ratio is greater than 1 for mesh and tree topology given some delay deadlines with R = 50kb/s. This is because, the proposed algorithm greedily finds the optimal energy consumption under the sub-deadline on the PCP but it only achieves the local optimum, which does not guarantee for the global optimum of the overall workflow scheduling.

(2) For case 2 under the LARAC algorithm, we also plot the energy ratio between the proposed workflow algorithm and the optimal solution with R = 50kb/s in Figure 4(b). The feasible set of the optimization problem is smaller due to the execution decision, and the energy ratio is much smaller.

(3) Moreover, for case 3, we also plot the energy ratio between the proposed workflow algorithm and the optimal solution in Figure 4(c). Since there is no solution for the general topology with R = 50kb/s, we present the results with R = 100kb/s. Also notice that there are no solutions for tree and general topology under some delays. These results in Figure 6 indicate that the proposed algorithm can achieve a good solution compared with the optimal solution.

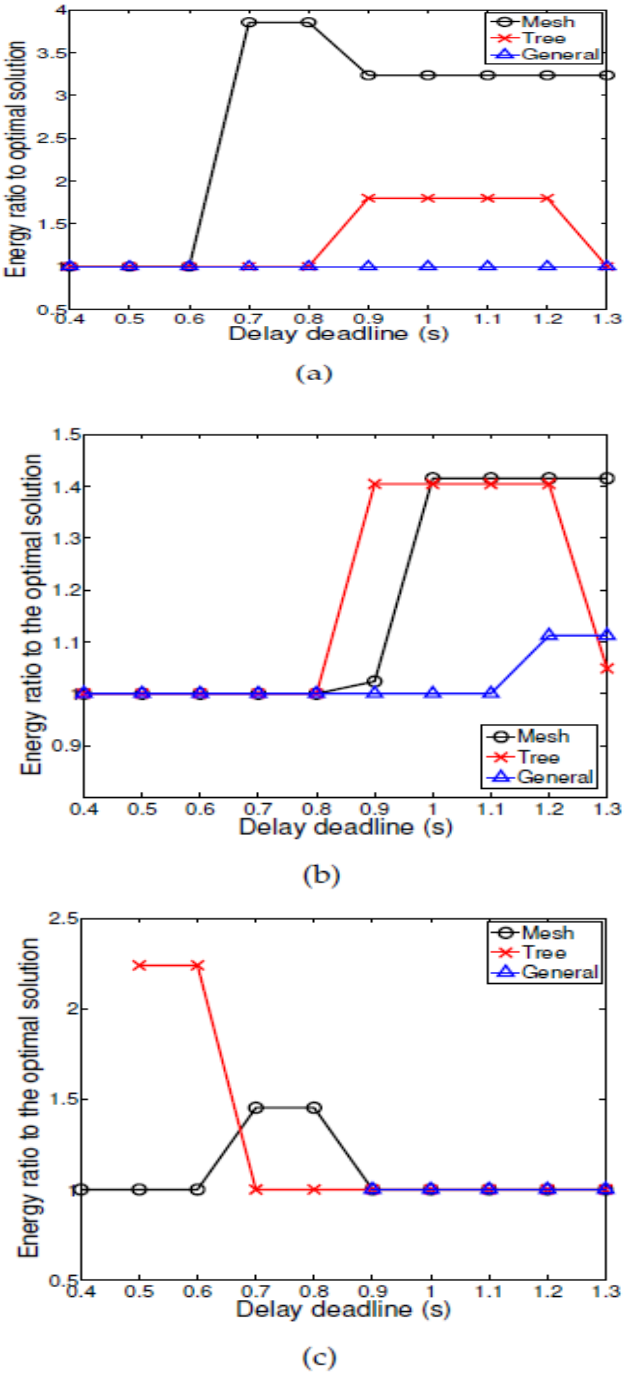

Fig. 4. Energy ratio between the proposed workflow scheduling algorithm and the brute-force enumeration.

(a) Case 1 as a special case under one-climb policy with R = 50kb/s. (b) Case 2 as a general case under LARAC algorithm with R = 50kb/s. (c) Case 3 as a general case under LARAC algorithm with R = 100kb/s.

CONCLUSION

In this paper, we proposed the collaborative task execution for the application as a general topology in mobile cloud computing. Each task within the general topology is either executed on the mobile device or on the

cloud clone. The objective was to minimize the energy consumption on the mobile device while meeting the delay deadline under the collaborative task execution. We formulated the collaborative task execution as a delay-constrained workflow scheduling problem. Based on the partial critical path analysis, we scheduled the tasks on the partial critical path. Specifically, we provided a general solution using LARAC algorithm to solve the general case with the consideration of execution restriction under which some tasks must be executed either on the mobile device or on the cloud clone. We also provided a solution using one-climb policy to solve the special case without the consideration of the execution restriction. Performance evaluation showed that the proposed workflow scheduling algorithm can obtain a reasonable solution compared with the brute-force enumeration. In addition, the collaborative task execution is more energy efficient and flexible than the local execution and the remote execution.

REFERENCES

[1]. Cisco Visual Networking Index: Global Mobile Data Traffic Forecast Update 2014 - 2019 White Paper, Cisco, 2015.

[2]. "How many apps are in the iphone app store?" last accessed on May 4, 2015. [Online]. Available: http://ipod.about.com/od/iphonesoftwareterms/qt/apps-in-app-store.htm

[3]. "Number of android applications," last accessed on May 4, 2015. [Online]. Available: http://www.appbrain.com/stats/number-of-android-pps

[4]. K. Kumar and Y. H. Lu, "Cloud computing for mobile users: Can offloading computation save energy?" IEEE Computer, vol. 43,no. 4, pp. 51–56, 2010.

[5]. H. T. Dinh, C. Lee, D. Niyato, and P. Wang, "A survey of mobile cloud computing: architecture, applications, and approaches,"Wireless communications and mobile computing, vol. 13, no. 18, pp.1587–1611, 2013.

[6]. Y. Cui, X. Ma, H. Wang, I. Stojmenovic, and J. Liu, "A survey of energy efficient wireless transmission and modeling in mobile cloud computing," Mobile Networks and Applications, vol. 18, no. 1, pp. 148–155, 2013.

[7]. E. Cuervo, A. Balasubramanian, D. Cho, A. Wolman, S. Saroiu, R. Chandra, and P. Bahl, "MAUI: Making smartphones last longer with code offload," in International conference on Mobile systems, applications, and services, 2010, pp. 49–62.

[8]. B. G. Chun, S. Ihm, P. Maniatis, M. Naik, and A. Patti,"Clonecloud: Elastic execution between mobile device and cloud," in Proceedings of the 6th European Conference on Computer Systems, 2011, pp. 301–314.

[9]. M. Satyanarayanan, R. C. P. Bahl, and N. Davies, "The case for vmbased cloudlets in mobile computing," IEEE Pervasive Computing, vol. 8, no. 4, pp. 14–23, 2009.

[10]. W. Zhang, A. Kunjithapatham, S. Jeong, and S. Gibbs, "Towards an elastic application model for augmenting the computing capabilities of mobile devices with cloud computing," Mobile Networks and Applications, vol. 16, no. 3, pp. 270–284, 2011.

[11]. G. Huerta-Canepa and D. Lee, "A virtual cloud computing provider for mobile devices," in Proceedings of the 1st ACM Workshop on Mobile Cloud Computing & Services: Social Networks and Beyond, 2010, p. 6.

[12]. N. Fernando, S. W. Loke, and W. Rahayu, "Dynamic mobile cloud computing: Ad hoc and opportunistic job sharing," in Proceedings of the Fourth IEEE International Conference on Utility and Cloud Computing, 2011, pp. 281–286.

[13]. W. Zhang, Y. Wen, K. Guan, D. Kilper, H. Luo, and D. O. Wu, "Energy-optimal mobile cloud computing under stochastic wireless channel," IEEE Transactions on Wireless Communications, vol. 12, no. 9, pp. 4569–4581, 2013.

[14]. W. Zhang, Y. Wen, and D. O. Wu, "Collaborative task execution in mobile cloud computing under stochastic wireless channel," IEEE Transactions on Wireless Communications, vol. 14, no. 1, pp. 81–93, 2015.

[15]. L. Yang, J. Cao, Y. Yuan, T. Li, A. Han, and A. Chan, "A framework for partitioning and execution of data stream applications in mobile cloud computing," ACM Sigmetrics Performance Evaluation Review, vol. 40, no. 4, pp. 23–32, 2013.

BIOGRAPHIES

J. Manikandan, working as Asst. Professor in St. Martins Engineering college, Secunderabad. Completed M.E in CSE and B.E degree in CSE, current research interest in the area of Cloud computing.

A. Srinish kumar, working as Asst. Professor in St. Martins Engineering college, Secunderabad, and pursuing Ph.D ,completed M.Tech in CSE at SCCE (JNTUH) (2009-12), and B.Tech degree in CSE from JNTUH, current research interest in the area of Data mining, AI,Cloud computing.

Goddeti Mallikarjun, working as Asst. Professor in St. Martins Engineering college, Secunderabad. Completed M.Tech in CSE, and B.Tech degree in IT from JNTUH, current research interest in the area of Artificial Intelligence.