2014 IEEE 6th International Conference on Cloud Computing Technology and Science

Ad-hoc Cloudlet Based Cooperative Cloud Gaming

Fangyuan Chi, Xiaofei Wang, Wei Cai, Victor C. M. Leung
Department of Electrical and Computer Engineering
The University of British Columbia
Vancouver, Canada
Email: {fangchi, xfwang, weicai, vleung}@ece.ubc.ca

***Abstract*—As the game industry matures, processing complex game logics in a timely manner is no longer an insurmountable problem. Many researchers are now trying to find ways to optimize the gaming system regarding the network usage, local resource utilization, and energy consumption. However, current cloud-based mobile gaming solutions are limited by their relatively high requirements on Internet resources. Also, they typically do not consider the geographical locations of nearby mobile users and thus ignore the potential cooperation among users. Therefore, inspired by existing cloud computing techniques and the concept of ad-hoc cloudlet computing, in this paper, we propose an ad-hoc cloudlet based gaming architecture. Two modules of the architecture are introduced: 1) progressive game resources download, by which mobile users can adaptively download gaming resources from cloud servers or nearby mobile users according to the gaming progress, and 2) ad-hoc cloudlet-based cooperative task allocation, by which gaming components can be executed dynamically over local devices, nearby devices, or cloud servers. We also formulate the mechanism for both modules as an optimization problem and propose several algorithms for both modules, which are later used for evaluation purposes. We carry out simulations based on real mobility traces, and the results show that our system's performance depends highly on the ad-hoc network environment (the more concurrent and balanced connections within the ad-hoc network, the lower the energy costs). Also, regardless of the network environment, our system has lower energy costs while utilizing resources of nearby devices, compared to the cloud-based gaming architecture.**

***Index Terms*—Cloud-let, Cooperative, Ad-hoc, Mobile Gaming.**

.

I. Introduction

[1] With Internet connectivity, mobile games can be played virtually anywhere anytime, which provides a fresh user experience to game players. However, hardware constraints imposed by mobile devices such as limited processing capabilities, storage capacities, and relatively short battery lives compromise the Quality of Experience (QoE) of the mobile gamers. To overcome these limitations, researchers have started to consider building game applications upon the concept of Mobile Cloud Computing (MCC) [1]. This new gaming architecture, called Mobile Cloud Gaming (MCG) inherits both the advantages and the challenges of MCC, e.g., unreliable network connectivity, limited network bandwidth, and unpredictable network latency [2]. More specifically, as indicated in [3], video-based cloud gaming uses a thin-client approach which offloads computation-intensive tasks to the cloud and thus addresses the aforementioned hardware constraints of mobile devices. However, computation-intensive tasks, combined with the need to render, encode and transmit the game scene in near real-time to the currently connected players make both server and Internet availability the most crucial aspects in the design of this type of system.

Aside from video-based cloud gaming architecture, designs based on workload offloading are proposed. For instance, companies such as Kalydo[2], Approxy[3] and SpawnApps[4] consider a new business model, called file-based cloud gaming. In this newly developed model, segmented game resources, including game data (e.g., textures, geometry, objects, and non-player character (NPC) metadata) and game logic (e.g., game rules, scripts, rendering, and network access), are progressively downloaded onto mobile devices. Games are then executed on-device with only a fraction of the total game resources downloaded (e.g., less than 5% of the game). Without the necessity to transmit video frames in real-time, less network bandwidth is consumed compared with the conventional cloud gaming architecture. However, the QoE is still limited by mobile devices' resource constraints. Considering the diversity of the execution and application environments, a component-based cloud gaming architecture [4] is proposed. It has been said that computer games are essentially constructed by a set of components, which are inter-dependent functions or objects that facilitate the logics of the games, and these components can migrate to mobile devices or cloud in response to different network and execution environments. This architecture partitions game application dynamically so that resources of both the mobile devices and the cloud are utilized. However, it still relies heavily on network connectivity and the performance depends on the network latency.

Both approaches do not consider the potential cooperation among nearby mobile devices. More specifically, enabling resources of nearby mobile devices, namely, their processing as well as storage capacities, to be pooled together to form an ad-hoc cloudlet, computation tasks could be offloaded to the ad-hoc cloudlet instead of the cloud. This mechanism, called ad-hoc cloudlet computing [2] [5] [6], provides an

[1]This work is supported by a University of British Columbia Four Year Doctoral Fellowship and by funding from the Natural Sciences and Engineering Research Council under Grant STPGP 447524.

[2]www.kalydo.com
[3]www.approxy.com
[4]www.spawnapps.com

978-1-4799-4093-6/14 $31.00 © 2014 IEEE
DOI 10.1109/CloudCom.2014.112

advantage of having less network latency and lower energy costs (i.e., less energy consumed by mobile devices for data transmissions and device discovery) as compared to cloud computing since mobile devices could communicate with each other via device-to-device (D2D) connections [7]. However, it has been identified in [5] that the task success rate and execution speed of ad-hoc cloudlet computing depend on the cloudlet access probability. Thus, the question is *could we use ad-hoc cloudlet computing in the cloud gaming architecture?* Ideally, the answer is yes: as measured in [11], game players tend to stay in the same place, e.g., public facilities, public transportation, and residential areas, for an extended period of time while playing games, the probability of a game player needing to access a new ad-hoc cloudlet during a game session is low. Thus, we assume that in the local level there is a stable D2D connectivity that allows the local cloudlet to be established.

Thus, in this paper, we target the above-mentioned constraints by considering an ad-hoc cloudlet based gaming architecture, interpolating both the file-based and component-based cloud gaming architectures. Our contribution is a system that allows users to share downloaded game data with each other and enables computation tasks to be dynamically distributed amongst users while taking different game progression into account. Moreover, our system enables users to offload tasks to the cloud when the ad-hoc cloudlet lacks the processing power for all the computation tasks. The main benefits of our approach are the reduced data transmissions between the users and the cloud due to ad-hoc cloudlet collaborations, the minimized redundant downloading, the minimized overall energy costs, and the maximized ad-hoc cloudlet resource utilization.

The rest of the paper is organized as follows. We first study some related works in Section II. Sections III and IV, respectively, introduce details of our proposed architecture, formulate both progressive and task allocation process as optimization problems, and propose several approximation algorithms such as the greedy algorithm, which have low complexity and near-optimal performance. Section V shows the simulation results and Section VI concludes the paper.

II. Related Work

Applications developed using the component-based programming model are composed of independent modules (components) which can be dealt with separately by multiple cloud services. In [8], an adaptation approach is proposed for multi-cloud applications, which allows dynamic changing of services for each component during run time without user intervention. Similarly, a cognitive platform has been designed [9] for modularized gaming applications, which enables computation task migration and dynamic task allocation between the cloud and mobile devices. Both works show that the component-based programming model is feasible and promising for the cloud computing paradigm. However, they suffer from the limitation that they have yet to take into consideration both the costs of data and task migration, and the unpredictable response latency when deciding to migrate from one service to another.

With the increasing processing capacity of modern mobile devices, cloudlet computing is an emerging topic that is attracting growing attention in the research community. Previous research works [10] have explored the possibility of deploying cloudlet-assisted multiplayer cloud gaming, especially to utilize cloudlet cooperative sharing mechanisms to optimize the system. While the system benefits from the reduced overall bandwidth consumption, it introduces additional latency due to the real-time encoding, sharing, and decoding mechanism. A gaming platform is proposed in [11], which enables interactive multi-player gaming with distributed game operations amongst users. In this platform, data packets are overheard by all participants, which result in unnecessary expenditure of battery energy of the recipients. The architecture proposed in [12] allows mobile game users to offload computation-intensive tasks to a system component called proxy client, which resides in the cloud and is responsible for decoupling the creation of rendering instructions from their execution and transmitting only the rendering instruction to mobile devices. Although the network bandwidth usage is reduced, the system performance depends highly on the cellular network connectivity, which may be intermittent and costly. Actually, such limitations can be reduced if the applications are deployed over an ad-hoc network, since direct connections using Wi-Fi within an ad-hoc network has much better network connectivity than external connections [13].

Various applications have been proposed for taking advantage of the mobile ad-hoc networks (MANET), e.g., ad-hoc streaming [14], which allows mobile users to download video collaboratively. Other applications such as cooperative downloading [15] and ad-hoc caching [16] all use similar ideas, i.e., to distribute computational tasks to members of a MANET. Also, cloudlet computing over ad-hoc network has become popular in the computing industry. In the decentralized Mobile Cloud Computing Server architecture [17] for ad-hoc networks, each mobile device acts as a server to support cloud computing services. In [18], a real-time face recognition application using mobile-cloudlet-cloud architecture is introduced; it is shown that the network response time can be minimized with the assistance of an ad-hoc cloudlet.

III. Framework Details

A. Architecture

In the proposed architecture, we consider the situation where a group of users would like to play the same game together within the same ad-hoc network. Together they form an ad-hoc cloudlet and all of the them simultaneously connect to both the cloud and the ad-hoc cloudlet. This application scenario is illustrated in Fig. 1, where the ad-hoc cloudlet is formed by seven users, each of whom performs two actions simultaneously: progressive downloading of segmented game resources and collaborative task allocation. In progressive downloading, users are assumed to hold some initial states of the game, and all game data as well as game components are progressively

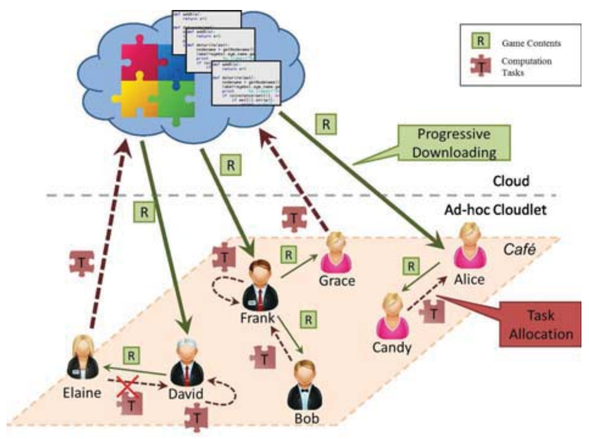


Fig. 1. Cooperative Ad-hoc Gaming

downloaded onto their mobile devices. As only a small number of game resources are required for users to start playing the game, the rest of the resources can be acquired from either the cloud or the neighbourhood ad-hoc cloudlet depending on the progress of the game. In the example, only users named David, Frank and Alice are getting game resources from the cloud and other users are all getting resources from their neighbours. Also, our system prioritizes the ad-hoc cloudlet when offloading the computation tasks, and only offloads to the cloud when the ad-hoc cloudlet does not have the required capability. Upon receiving the task allocation requests, users could decide to either accept or reject the requests based on their current processing status. In the example, four task allocation cases are included: 1) Alice does not have a local task so she can process the task for Candy; 2) Grace has a task that cannot be handled by the ad-hoc cloudlet and hence offloads her task to the cloud; 3) Frank is processing both his local task and the task offloaded by Bob; 4) Elaine's request to offload her task to David is rejected since David is busy processing his own task; thus Elaine offloads her task to the cloud. Also, game states and user generated data are synchronized between users and back to the cloud.

We focus on progressive downloading and task allocation processes since they are the determining factors for our system performance. Given that these two processes depend on the speed of users' game progresses, a global knowledge of users' game statuses should be maintained by each user and updated in real-time. To achieve this, each user is required to periodically send a beacon so that all users are aware of their one hop neighbours and also their game statuses. In these models, we consider two transmission schemes: 1) transmissions between users and the cloud over a cellular (3G/4G) network; 2) transmissions within the ad-hoc cloudlet via D2D connectivity. Each scheme has its own characteristics in terms of energy costs. For instance, the ramp and tail energy are results of the Radio Resource Control (RRC) protocol in 3G networks, which are not present with Wi-Fi. Similarly, Wi-Fi has a high association cost and device discovery cost. In this paper, we focus on their common component, which is the energy cost for data transmissions. This energy cost is proportional to the size of transmitted packets and the transmission time. It is measured and modeled in [19] which shows that data transmissions via D2D links consume less energy. Moreover, to determine the energy costs for offloading to the ad-hoc cloudlet, each time the beacon message is received, energy costs between its transmitter and receiver are estimated using the received signal strength (for a poor signal strength, higher energy is consumed) [20]. Therefore, aiming at reducing energy costs, it is critical for the system to determine to whom to map downloading requests and computation tasks to. Also, for simplicity, we allow only unicast transmissions within the ad-hoc cloudlet since offloading to the cloud uses one-to-one transmissions.

We will formulate the downloading and task-resource mapping mechanism as an optimization problem in the next section. Hereby we define an N by N **neighbourhood matrix**, X, where each element represents relationship between two users, to model users in an ad-hoc cloudlet. $X_{ij} = 1$ indicates that there exists a direct communication between user u_i and u_j and they will be able to collaborate. Also, to enable tasks to be handled locally, X_{ii} should always be 1.

B. Progressive Downloading of Gaming Resource

In this module, we denote game resources downloaded directly from the cloud as $d_i^{3G/4G}(t)$, and game resources acquired from the ad-hoc cloudlet as $d_i^{D2D}(t)$. At time t, each user u_i will have its cellular link speed $s_i(t)$ in MB/s, current downloaded resources $d_i(t)$, gaming process speed $g_i(t)$ and current game progress $G_i(t)$. Then, the overall downloading can be formulated as:

$$\sum_{i=1}^{N}\sum_{t=0}^{t=T_i} d_i^{3G/4G}(t) + d_i^{D2D}(t) \tag{1}$$

Our main purpose is to minimize the downloaded contents via cellular link, which can be formulated as:

$$\text{Minimize:} \quad \sum_{i=1}^{N}\sum_{t=0}^{t=T_i} d_i^{3G/4G}(t) \tag{2}$$

C. Cooperative Task Allocation for Ad-hoc Cloudlet Gaming

We define two types of energy costs for computation tasks, each type represents the energy costs for one of the aforementioned transmission schemes. We denote $cost_{ij}(t)$ as the costs for transmitting the task via Wi-Fi (offload to the cloudlet), and $cost_{ic}(t)$ as the costs for transmitting the task via 3G/4G (offload to the cloud). Generally, we have $cost_{ij}(t) \le cost_{ic}(t)$; thus, to reduce the overall costs, we want to map as many tasks to the ad-hoc cloudlet as possible. The task allocation mechanism could be considered as a multiple knapsack problem, where the mobile device is considered as a knapsack with a capacity determined by its CPU processing power and storage capacity. The goal of this knapsack problem is to select N disjoint subsets of tasks based on the network topology, as it can only be mapped to its initiator's direct neighbour, which minimizes the total costs for the selected tasks. Each subset of tasks can then be assigned to different mobile devices (where the assigned tasks were initiated by the device's neighbours) whose CPU and storage capacity is no less than the total required CPU and storage capacity of tasks in the subset. The availability of mobile devices should be captured and maintained by real-time monitoring. At time t, user u_i has 1) a task $t_i(t)$, its storage consumption, $s_i(t)$ and CPU consumption, $c_i(t)$; 2) a computation potential, which is associated with availability, $A_i(t)$, available storage, $S_i(t)$ and available CPU load, $C_i(t)$. Since each computation task is mapped to either the ad-hoc cloudlet or the cloud, the costs for task allocation has two parts, which is the costs for offloading to the cloudlet and the costs for offloading to the cloud.

We define the total costs for offloading to cloudlet $COST_l$ as:

$$COST_l = \sum_{i=1}^{N}\sum_{j=1}^{N} F_{ij}(t) * X_{ij} * Cost_{ij} * A_j \tag{3}$$

We define the total costs for offloading to cloud $COST_c$ as:

$$COST_c = \sum_{i=1}^{N}(1 - \sum_{j=1}^{N} F_{ij}(t)) * Cost_{ic} \tag{4}$$

where

$$F_{ij}(t) = \begin{cases} 1, & \text{If user i's task is mapped to user j} \\ 0, & \text{Otherwise} \end{cases} \tag{5}$$

Then, we define the costs $COST$ for task allocation as:

$$COST = COST_l + COST_c \tag{6}$$

To minimize the overall costs, in general, we have:

$$\begin{aligned} \text{Minimize:} \quad & COST \\ \text{Subject to:} \quad & \sum_{i=1}^{N} F_{ij}(t) * X_{ij} * A_j(t) * s_j(t) \leq S_j(t), \forall j, \\ & \sum_{i=1}^{N} F_{ij}(t) * X_{ij} * A_j(t) * c_j(t) \leq C_j(t), \forall j, \\ & \sum_{j=1}^{N} F_{ij}(t) * X_{ij} * A_j(t) \leq 1, \forall i, \\ & F_{ij}(t) \in \{0,1\} \end{aligned} \tag{7}$$

Four constraints are defined in the formulation. The first and the second are used to indicate that the overall CPU and storage consumption for the assigned tasks to each user is no more than the CPU and storage available for such user. The third constraint is to assign no more than one task to each user, and the last indicates that for $F_{ij}(t)$ we only accept solutions as 0 or 1 as defined earlier.

IV. Algorithms

In this section, we propose several approximation algorithms for finding optimal solutions. Since computer games are delay-sensitive applications, finding the exact optimal solution may not be feasible as it may take too long. Also, the allocation process for each task and the downloading request for each user can be considered as a sub-problem to a global optimization problem. Thus, for each sub-problem, we can iterate through the feasible solutions and apply a heuristic to determine a near-optimal solution, where the optimal solution is defined by having the minimal processing time. In the following section, heuristics algorithms for both modules are presented, which could be applied under different network environment to gain better system performance.

A. Progressive Download of Gaming Resource

For this module, we consider two heuristic algorithms. The first one is called greedy local, which allows users to acquire game resources from a neighbour with the minimal transmission speed. This algorithm is summarized in Algorithm 1, where a user u_i needs a game resource that is owned by a user u_j, it will then estimate the time for the owner to transmit the content. This time will be denoted as $T_{ij}(t)$. Finally, it will retrieve its gaming progress speed, $g_i(t)$, and decide whether to receive the content through the server or its neighbour by comparing $T_{ij}(t)$ and $g_i(t)$: if $T_{ij}(t) < g_i(t)$, the user should be receiving the content through its neighbour; otherwise, the content is downloaded from the cloud.

Algorithm 1 Algorithm: Greedy Local

```
1:  $U$ : Set of neighbours who has the required game piece
2:  $g$ := Local game progress speed
3:  $D := \emptyset$
4:  while $U \neq \emptyset$ do
5:      choose $u \in U : T_u$ being minimal and less than $g$;
6:      if $u \neq Null$ then
7:          $G \leftarrow downloadFromNeighbour(u)$
8:      else
9:          $G \leftarrow DownloadFromCloud(G)$
10:     end if
11:     $D := D \cup \{G\}$;
12: end while
```

Algorithm 2 Algorithm: Weighted K-Mean

```
1:  $D := \emptyset$
2:  $W$ := Set of weighted users
3:  $K$ := Set of user being center to each cluster
4:  $K \leftarrow KMeanClustering(W)$;
5:  for $i \leftarrow 0$ to $N$ do
6:      if $W_i \in K$ then
7:          $G \leftarrow downloadFromCloud(G)$;
8:      else
9:          $G \leftarrow downloadFromNeighbour()$;
10:     end if
11:     $D := D \cup \{G\}$;
12: end for
```

Then, a different strategy is used for which users are organized into clusters and can only acquire game resources from users within the same cluster. Each user u_i is associated with a weight, G_i, which is the current game progress for u_i. For each time interval, the K-Means algorithm for clustering is applied to find the best K users who are responsible for downloading via cellular links. Other users will primarily receive the required content from these K users. For the situation where user u_i is no longer satisfied acquiring resource from its peers, it can decide to download contents via a cellular link on its own. This algorithm is shown in Algorithm 2.

Both algorithms have computational complexity of O(1), assuming the K-Means clustering algorithm is running on the cloud and therefore we ignore its associated complexity. These algorithms have different advantages and disadvantages. The first algorithm allows each user to select neighbours freely and distributes the downloading requests evenly within the ad-hoc cloudlet, whereas the second algorithm only allows users to acquire data from system determined users, which is not as fair as the first algorithm. However, the first algorithm has a higher requirement in terms of user reliability since all users are responsible for sharing their acquired data while the second algorithm allows users to only acquire contents from determined users. Also, the second algorithm works better when some of the users have much faster game progress speed as compared to the rest of the users since the users who are responsible for downloading via cellular links are determined according to their game speed.

B. Cooperative Task Allocation for Ad-hoc Gaming

The basic structure of the algorithm for this module is presented in Algorithm 3. Assuming there is a mapping M, which represents the task-resource mapping within the ad-hoc cloudlet, and we have to specify a set of tasks initiated by users within the ad-hoc cloudlet and decide to which neighbour the task is allocated so that the energy costs are minimized. We use several different strategies for choosing the neighbour to offload the task, where they all have to meet one constraint, which is for the chosen neighbour to have sufficient computation potential for the task. For example, if user u_i has a task T_i and wants to map to its neighbour u_j, then user u_j must have $S_j \geq s_{T_i}$ and $C_j \geq c_{T_i}$. If no valid neighbour is in the cloudlet, the task is then offloaded to the cloud.

- **Random Mapping**: Task requests are mapped to randomly chosen neighbours who have enough processing and storage capabilities.
- **Greedy Local**: Each potential task-resource mapping has its corresponding communication task. In the previous section we used $Cost_{ij}$ to refer to costs for offloading task from user u_i to u_j. In this strategy, each task is assigned to the available neighbour with minimum $Cost_{i,neighbour}$, thus, the globally optimal solution can be achieved by making locally optimal choices.
- **Greedy Heuristic**: In this strategy, each task is assigned to user u_j if it has the minimum additional $Cost_{i,j}$ based on its current mapping.

Algorithm 3 Algorithm Frame for Task Allocation
```
1: $T$ := Set of Tasks
2: $M$ := $\emptyset$
3: for $i \leftarrow 0$ to $N$ do
4:     $choose\{u \in neighbourOf(initiator(T_i))\}$;
5:     if $u \neq Null$ then
6:         $M := M \cup \{T_i, u\}$;
7:         $C_u := C_u - c_{T_i}$;
8:         $S_u := S_u - s_{T_i}$;
9:     else
10:        $OffloadToCloud(t)$
11:    end if
12: end for
```

TABLE I
COMPLEXITY

Algorithm	Computational Complexity	Storage Complexity
Random Mapping	O(1)	O(1)
Greedy Local	O(n)	O(n)
Greedy Heuristic	O(n)	O(n^2)
Greedy Cluster	N/A	N/A

- **Greedy Cluster**: Users are organized in groups, and tasks are mapped to users within the group using any strategy presented above.

The computational and storage complexity for the above algorithms are listed in Table I. In our analysis, we consider only the complexity for the neighbour selection process since steps for maintenance are common for all algorithms.

V. Evaluations

System performance is evaluated using a Java simulator, which assigns random attributes to users and computation tasks. Associated with each user, several parameters are defined such as its game progress speed, storage availability, CPU availability and cellular link speed, whereas for computation tasks, parameters such as storage consumption, CPU consumption, its costs for offloading to the cloud as well as the costs for offloading to the ad-hoc cloudlet are defined. Each of these parameters is randomly generated within a certain range, which is specified using base cases and different factors. To better test our system performance, the users' CPU availability has a range smaller than tasks' CPU consumption, so that some tasks will have CPU consumption greater than the ad-hoc cloudlet's processing capability and hence will have to be offloaded to the cloud. Details of how each parameter is generated are listed in Table II.

We then simulated our system using randomly generated network topologies, which allows us to control its associated density and skewness, and network topologies generated based on real-world traces and users' movement model. Simulation results are shown in the following sections.

A. Randomly Generated Network Topology

1) Progressive Downloading: Each user downloads a game resource depending on their game progress speed, which is the

TABLE II
SETUP PARAMETERS

Target	Parameter	Minimum	Maximum
Base Case	CPU	50	350
	Storage	50	350
Users	Game Progress Speed	1	10
	Storage Availability Factor	0.9	1
	CPU Availability Factor	1	1
	Cellular Link Speed	1	5
Tasks	Storage Consumption Factor	0.2	1.05
	CPU Consumption Factor	0.7	1.15
	Cost to cloud	1	10
	Cost to local	4	40
Network Topology	Density	1	10
	Skewness	1	10
	Task Density	1	10

time slots interval between each download. An example of the downloading progress for ten users is shown in Fig. 2. In this example, game data are segmented into ten pieces and the downloading process is implemented using Algorithm 1. The experimental results in Fig. 2 show that users with faster game progression speed finishes downloading before others.

As game resources can be acquired either from neighbours within the ad-hoc cloudlet or the cloud, we differentiate the downloaded game pieces via different network links. Fig. 3 shows the total downloaded game data for all users. Game pieces downloaded directly from the cloud (via cellular links) are coloured in blue whereas pieces acquired from cloudlet (via D2D links) are colored in red. From the graph, we can see that at the beginning of the game session, most of the game data are downloaded from the cloud; however, as the game progresses, more and more game data are acquired through the ad-hoc cloudlet. Thus, total downloading traffic via cellular link is significantly reduced.

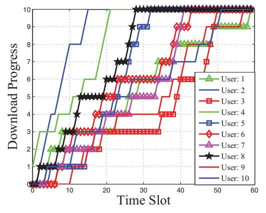


Fig. 2. Download Progress

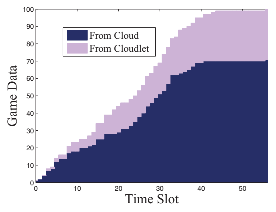


Fig. 3. Total Download

2) Task Allocation: We simulated and evaluated the performance for a single time slot under different randomly generated network environments. Several algorithms are used to implement the task allocation process, including the solution to the optimization problem formulation, which is produced using optimization solver called Gurob, and their results are compared side by side. We included only the CPU and storage consumed by computation tasks and ignored the resources used for carrying out each algorithm in the simulations.

- **Density**: Fig. 4 shows the overall energy costs incurred by different allocation algorithms under different network environments. The network topology is generated in response to different degrees of network density, where a topology with higher density indicates that there is more D2D connectivity between users than a topology with lower density. These experimental results show that the energy costs decrease as the network density increases for all the algorithms. This is because a higher degree in density means users have more connected neighbours, which implies that they have more external helpers so that more computation tasks can be offloaded to the ad-hoc cloudlet, and the costs to offload tasks to the ad-hoc cloudlet are often smaller than the alternative. Fig. 8 and 6 shows the idle storage and CPU capacity in percentage with various degree in density. From both figures, we see that idle resources decrease as density increases, which shows that our system gives better resource utilization. Also, we observe that the greedy heuristic algorithm has lower energy costs in the beginning but converges with the greedy local algorithm as density level increases, which indicates that in a more connected network, the greedy local algorithm has lower energy costs.
- **Skewness**: In fact, to try to simulate the system and analyze the system performance in a real-world environment, we need to obtain a more realistic network topology, which follows the power law. Thus, we used an algorithm proposed in [21] to generate such a network topology. Then, we obtained the energy costs under various degrees of skewness, where a high degree in skewness means that the network is more unbalanced (i.e., only a small number of users are connected to most other users while others are connected to only a few) than low degree. From the experimental results shown in Fig. 5, we can see that with a higher degree in skewness, the overall energy costs increases. This is because with unbalanced network topology, most of the users have few connected neighbours while only a small number of users have most of the connections. Sometimes users with few neighbours will have to offload their computation tasks to the cloud; thus the energy costs are high. This indicates that with a more balanced network topology, the system costs are minimized. Also, our system has better resource utilization when the skewness is minimal, as showed in Fig. 9 and 7. We observe that the greedy heuristic algorithm has higher energy costs in the beginning but lower costs as skewness level increases as compared to the greedy local algorithm, which indicates that in a more balanced network, the greedy local algorithm has lower energy costs.
- **Density and Skewness**: A more extensive experiment is done for the allocation process, taking into account both skewness and density when generating the network topology. The experimental results are shown in Fig. 10, which evidently prove that an unbalanced and unconnected (i.e., most users do not have connected neighbour) network topology causes higher energy costs.

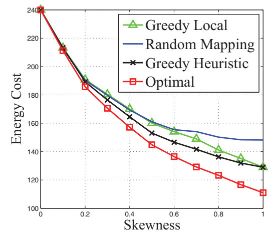


Fig. 4. Costs for Various Density

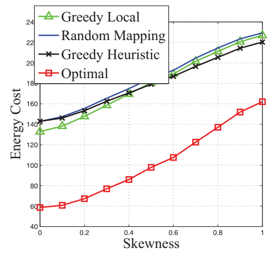


Fig. 5. Costs for Various Skewness

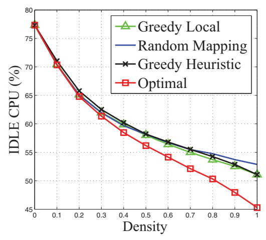


Fig. 6. CPU Utilization with Various Density

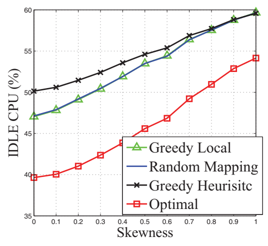


Fig. 7. CPU Utilization with Various Skewness

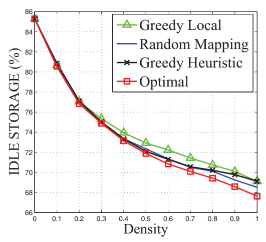


Fig. 8. Storage Utilization with Various Density

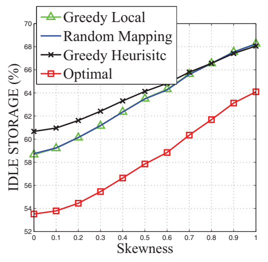


Fig. 9. Storage Utilization with Various Skewness

B. *Real-world Trace-based Network Topology*

The system is simulated using an open-source network topology generator called ONE[5], which is based on real-world traces and users' movement model. In this section, two types of movement patterns are used, namely, random walks, and map based movements. In random walks, users' locations are randomly generated and users move in random patterns regardless of their geographic locations, whereas in map-based movements, users are located inside several blocks initially and their movement patterns are according to the map of each block. Examples of network topologies generated based on different movement patterns are shown in Fig. 11 and 12 respectively.

Energy costs incurred by our task allocation module are captured for different task density, which is the probability of users having tasks to offload at each time slot. Experimental

[5]www.netlab.tkk.fi/tutkimus/dtn/theone

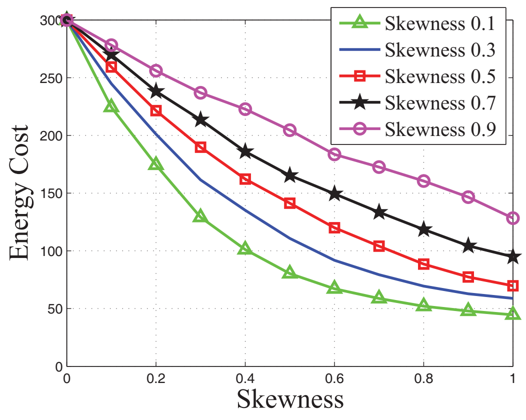
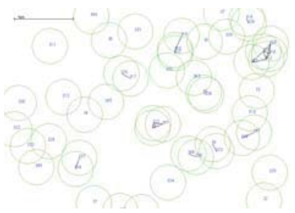


Fig. 10. Costs for Various Skewness and Density



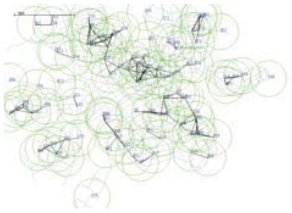
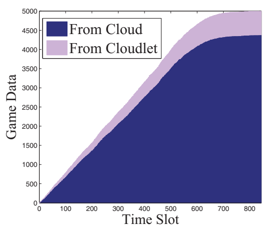


Fig. 11. Network Topology with Random Walk Movement Pattern

Fig. 12. Network Topology with Map-Based Movement Pattern

results for random walk, and map based movement topology are shown in Fig. 14, 13, 15 and 16 for both progressive downloading and task allocation. From these figures, we can see that as a game progresses, more and more game data are acquired from the ad-hoc cloudlet and total traffic via the cellular network is reduced. Also, energy costs incurred by our proposed algorithms increase with task density but much slower than energy costs incurred without our algorithm. Among all the algorithms, the greedy local algorithm has lower energy costs when the network has a more regular mobility pattern.



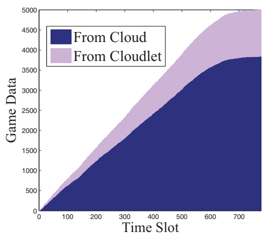


Fig. 13. Progressive Downloading with Random Walk

Fig. 14. Progressive Downloading with Map-Based Movement

All experimental results presented above show that the system performance depends highly on the network environment, and more benefits are realized by the proposed scheme in a balanced and connected network. However, regardless of the network environment, the algorithms we have proposed

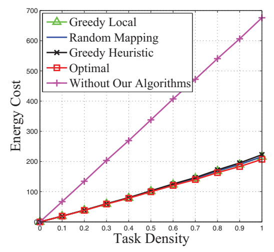


Fig. 15. Task Allocation with Random Walk

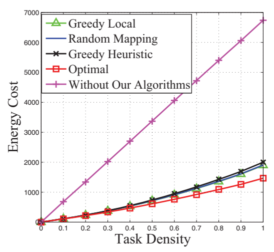


Fig. 16. Task Allocation with Map-Based Movement

are shown to be near-optimal and hence are cost-efficient. More specifically, the greedy local algorithm performs better when the network topology is more connected, balanced, and has a more regular mobility pattern. In contrast, for less dense and unbalanced network, the greedy heuristic algorithm would be more cost-efficient. Also, random mapping could be considered as a good alternative when computation and storage capacity of mobile devices are limited, since it has lower computational and storage complexity as indicated in Table I.

VI. Conclusion

In this paper, we have proposed an ad-hoc cloudlet based gaming architecture, which considers both progressive and collaborative downloading of game resources as well as co-operative task processing. Simulations of our framework have been done using several proposed algorithms under different network environment settings. The results show that our proposed algorithms have lower energy costs compared to those without our framework. Also, different algorithms could be applied to obtain a better system performance; e.g., in a more balanced, connected network environment, the greedy local algorithm tends to minimize the energy costs.

Regarding future work, we are planning to investigate the effects of task duration and users' mobility patterns, as well as the probability of packet loss. Moreover, the main overhead for both modules are the beacon messages sent, as well as the memory used to acquire and store the neighbours' gaming statuses. As they all happen within the local ad-hoc network, their impact on performance is minimal. Thus we will ignore this impact at the time and measure the overhead in the future. The scalability of this system is acceptable, since both progressive downloading and cooperative task processing are decentralized decision-making processes. However, efficient and decentralized service discovery, device discovery, and membership management mechanisms should be carefully designed to ensure the scalability of the system, which will also be considered as future work.

References

[1] H.T. Dinh, C. Lee, D. Niyato, P. Wang, "A survey of mobile cloud computing: Architecture, applications, and approaches," Wireless Communications and Mobile Computing (WCMC), vol. 13, no. 18, pp. 15871611, December 2013.

[2] C. Zhu, V. C. M. Leung, X. Hu, L. Shu, and L. T. Yang, "A review of key issues that concern the feasibility of mobile cloud computing," In Proc. IEEE CPSCom, pp. 769-776, 2013.

[3] C. Huang, C. Hsu, Y. Chang and K. Chen, "GamingAnywhere: An open cloud gaming system", in *ACM Multimedia Systems Conference*, Oslo, Norway, Feb./Mar. 2013, pp. 36-47.

[4] W. Cai and V. Leung, "Decomposed Cloud Games: Design Principles and Challenges", in *IEEE International Conference on Multimedia and Expo (ICME2014)*, Chengdu, China, July 2014.

[5] Y. Li and W. Wang, "The unheralded Power of Cloudlet Computing in the Vicinity of Mobile Devices", in *IEEE Global Communications Conference (Globecom' 13)*, Atlanta, GA, USA, 9-13 December, 2013.

[6] R. Kaewpuang, D. Niyato, P. Wang, E. Hossain, "A framework for cooperative resource management in mobile cloud computing," IEEE Journal on Selected Areas in Communications - Special Issue on Networking Challenges in Cloud Computing Systems and Applications, vol. 31, no. 12, pp. 2685-2700, December 2013.

[7] E. Koukoumidis, D. Lymberopoulos, K. Strauss, J. Liu, and D. Burger, "Pocket cloudlets", in *Sixteenth International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, Mar. 2011.

[8] A. Almeida, E. Cavalcante, T. Batista, N. Cacho and F. Lopes, "A Component-Based Adaptation Approach for Multi-Cloud Applications", in *IEEE INFOCOM Workshop on Cross-Cloud Systems*, Toronto, Canadda, April 2014.

[9] W. Cai, C. Zhou V. Leung and M. Chen, "A Cognitive Platform for Mobile Cloud Gaming", in *IEEE 5th International Conference on Cloud Computing Technology and Science (CloudCom2013)*, Bristol, UK, December 2013.

[10] W. Cai, V. Leung and L. Hu, "A Cloudlet-assisted Multiplayer Cloud Gaing System", in *ACM Springer Mobile Networks and Applications (MONET)*, vol. 19, No.2, pp. 144-152, April 2014.

[11] S. Pushp, C. Liu, F. Liu and J. Song, "Multi-Player Gaming in Public Transport Crowd: Opportunities and Challenges", in *IEEE World Forum on Internet of Things*, Seoul, Korea, March 2014.

[12] D. Meilander, F. Glinka, S. Gorlatch, L. Lin, W. Zhang and X. Liao, "Bringing Mobile Online Games to Clouds", in *IEEE INFOCOM Workshop on Mobile Cloud Computing*, Toronto, Canadda, April 2014.

[13] D. Camps-Mur, A. Garcia-Saavedra, and P. Serrano, "Device-to-device Communications with Wi-Fi Direct: Overview and Experimentation", in *IEEE Wireless Communications Magazine*, vol. 20, no. 3, pp. 96–104, June, 2013.

[14] K. Lorenzo, L. Anh and C. Blerim. "MicroCast: Cooperative Video Streaming on Smartphones", in *ACM MobiSys*, 2012.

[15] J. Wang, S. Wang, Y. Sun, W. Lu, D. Wu, "An incentive mechanism for cooperative downloading method in VANET", in *Vehicular Electronics and Safety (ICVES), 2013 IEEE International Conference*, on, vol., no., pp.125,130, 28-30 July 2013.

[16] B. Yang; A.R. Hurson and Y. Jiao, "On the Content Predictability of Cooperative Image Caching in Ad Hoc Networks," in *Mobile Data Management, 7th International Conference*, on , vol., no., pp.8,8, 10-12 May 2006.

[17] R. Agarwal and A. Nayak, "DRAP: A decentralized public resourced cloudlet for ad-hoc networks".

[18] T. Soyata, R. Muraleedharan, C. Funai, K. Minseok, W. Heinzelman, "Cloud-Vision: Real-time face recognition using a mobile-cloudlet-cloud acceleration architecture", in *Computers and Communications (ISCC), 2012 IEEE Symposium*, on , vol., no., pp. 59-66, 1-4 July 2012.

[19] N. Balasubramanian, A. Balasubramanian, and A. Venkataramani, "Energy Consumption in Mobile Phones: A Measurement Study and Implications for Network Applications," in *ACM Internet Measurement Conference*, Chicago, USA, 2009, pp. 280-293.

[20] N. Ding, D. Wagner, X. Chen, A. Pathak, Y.C. Hu, and A. Rice, Characterizing and modeling the impact of wireless signal strength on smartphone battery drain", in *ACM SIGMETRICS/international conference on Measurement and modeling of computer systems (SIGMETRICS '13)*, Jun. 2013, pp. 29-40.

[21] C.R. Palmer, J.G. Steffan, "Generating network topologies that obey power laws", in *Global Telecommunications Conference*, 2000. GLOBECOM '00. IEEE , vol.1, no., pp.434,438 vol.1, 2000.