Ad-Hoc Cloudlet Based Cooperative Cloud Gaming

Fangyuan Chi, *Student Member, IEEE*, Xiaofei Wang, *Member, IEEE*,
Wei Cai, *Member, IEEE*, and Victor C.M. Leung, *Fellow, IEEE*

Abstract—As the game industry matures, processing complex game logics in a timely manner is no longer an insurmountable problem. 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 them. Therefore, inspired by existing cloud computing techniques, we propose an ad hoc mobile-cloudlet-cloud based approach to implement cooperative gaming architecture. In this paper, 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, 2) ad-hoc mobile based cooperative task allocation, by which gaming components can be executed dynamically on local devices, nearby devices, stationary cloudlet(s), or cloud servers. The mechanisms of both modules are formulated as optimization problems and algorithms are proposed to solve them. Simulations results based on real mobility traces show that our system's performance depends highly on the ad-hoc network environment. Our scheme has lower system resource usage while utilizing resources of nearby devices, compared to the cloud-based gaming architecture; and performs better with short on-device task duration compared to code-offloading based architecture.

Index Terms—Cloud computing, cloudlet, cooperative, ad-hoc, mobile gaming

1 Introduction

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 the availability of both cloud services and the Internet the most crucial aspects in the design of this type of system. Also, current cloud-based solutions have several drawbacks: 1) current mobile networking solutions are often not energy-efficient [4] and has poor resource utilization; 2) by enabling worldwide players to request services from the distant cloud anytime and anywhere, cloud gaming services put a burden on cellular networks as it likely to be overloaded. To alleviate this, researchers have started to seek new mobile networking solutions, i.e., tiered networking solutions which allow potential cooperation among nearby users [5]; context-aware networked applications with converged heterogeneous wireless network access [6]; composite-radio infrastructure, which allows cellular and wireless local area network (LAN) to co-operate within a system to provide efficient wireless networking solutions, in terms of cost and Quality of Services (QoS) [7].

Inspired by these works, we consider providing mobility and context-aware cooperative cloud gaming services using tiered (*ad hoc mobile cloud-cloudlet-cloud*) network architecture. As mentioned in [8], resources of nearby mobile devices, namely, their processing as well as storage capacities, could be pooled together over an ad hoc network to form an **ad hoc mobile cloud**. Then, users could play games in a cooperative manner, where computation tasks are processed collaboratively on all neighboring mobile devices instead of solely on the centralized cloud. In addition, traditional cloud gaming architecture does not consider the possibility to deploy stationary **cloudlet(s)** [2] (i.e., a standalone processing unit that is deployed close to users) at places where users most likely to play games. Indeed, since people often play games together in coffee shops, and social gathering, the idea of embedding cloudlet(s) at places such as Wi-Fi Access Points (APs) and routers to assist mobile users seems promising. With this tiered networking architecture, we jointly optimize the utilization of resources from the ad hoc mobile cloud, the cloudlet(s), and the cloud under

- *The authors are with the Department of Electrical and Computer Engineering, The University of British Columbia, Vancouver, BC, Canada. E-mail: {fangchi, xfwang, weicai, vleung}@ece.ubc.ca.*

Manuscript received 31 Mar. 2015; revised 24 Sept. 2015; accepted 19 Oct. 2015. Date of publication 9 Nov. 2015; date of current version 31 Aug. 2018.
Recommended for acceptance by A. Vasilakos.
For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below.
Digital Object Identifier no. 10.1109/TCC.2015.2498936

2168-7161 © 2015 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission.
See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

Authorized licensed use limited to: The Chinese University of Hong Kong CUHK(Shenzhen). Downloaded on May 18,2020 at 02:48:48 UTC from IEEE Xplore. Restrictions apply.

several QoS constraints, i.e., energy consumption, bandwidth consumption, and interactive latency, under the condition that heterogeneous user demands exists. In fact, many researchers have been investigating resource allocation problem for cloud-based services, with respect to QoS optimization [9], [10]. Yet, different from these works, our architecture provides two types of cooperation in addition to the ability to offloads tasks to the cloud: ad hoc mobile cloud based cooperation with and without the assistance from the cloudlet(s), which is defined by the way in which mobile devices are connected to and interact with the cloud. To be more specific, our system provides progressive downloading, which allows users to share downloaded game resources with each other while taking different game progression into account. In addition, our system enables cooperative task allocation, which allows computation tasks to be 1) dynamically distributed amongst users within the ad hoc mobile cloud and the cloudlet(s); 2) offloaded to the centralized cloud when the ad hoc mobile cloud and the cloudlet(s) lacks the processing power for all the computation tasks.

The intention of enabling user-level cooperation is to provide a new gameplay experience with optimized QoS, in terms of reduced device-cloud bandwidth consumption, energy consumption of mobile devices, and the interactive latency. As a matter of fact, these three optimization objectives are rather conflicting, meaning that they could not be achieved individually without degrading the others. Thus, the system considers trade-offs when optimizing the performance: instead of simultaneously optimizing each objective, a set of weight factors is incorporated to reflect the relative importance of each objective. For each hosted games, the weight factors are initially assigned (by the system) to a default number, which is determined by several attributes of the game. Then, these factors could be altered by users to better suit their preferred gameplay experience. For instance, some users may care more about the interactive latency than the energy consumption. In this case, they could set the weight factors for the interactive latency higher than the energy consumption accordingly. The ultimate goal of the system is to provide a customized gameplay experience. From the system's perspective, the benefit gained from user-level cooperation is the reduced overall device-cloud data transmission and the optimized QoS. On the other side, from user's perspective, the benefit gained from user-level cooperation is the opportunity to access pooled resources by acting as a contributor and consumer at the same time, while having self-controlled and customized performance.

The rest of the paper is organized as follows. We first study some related works in Section 2. Sections 3, 4, and 5, respectively, introduce the 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 6 shows the simulation results and Section 7 concludes the paper.

2 Related Work and Preliminary

The ad hoc mobile cloud-cloudlet-cloud based architecture has many practical challenges. For instance, it has been identified in [11] that the task success rate and execution speed of cloudlet computing depend on the cloudlet access probability. Also, in ad hoc mobile cloud based computing, both the connection probability, which is the probability for users to connect as peers in the ad hoc mobile cloud; as well as the contact duration, which is the time duration they spend being neighbors, are key dimensions for the computation success rate. Thus, users' behavior and their mobility patterns need to be taken into consideration in such architecture. Then, the question is *could we apply the ad hoc mobile cloud-cloudlet-cloud architecture in the cloud gaming paradigm?* Ideally, the answer is yes: 1) as measured in [12], 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. Thus, with stationary cloudlet(s) deployed in such places, it is safe for us to assume that in the local level there is a stable device-cloulet connectivity; 2) as indicated in [13], social relationships between users could be used as a predictor to users' mobility pattern. Thus, we believe that by considering closeness in the task offloading process, the computation success rate could be ensured.

As mentioned and studied in [2], [14], and [15], with the assistance from the stationary **cloudlet(s)**, reduced interactive latency and communication costs could be achieved. Current studies suggest that cloudlet could be integrated with Wi-Fi Access Point, which could be considered as a physical integration of Wi-Fi AP and cloudlet. The most well studied method used to deploy cloudlet is hardware virtual machine technology [16], which has considerably low hardware and maintenance costs. The purpose/benefit of bring it into the hierarchy is to provide low-latency computation and rich computational resources. It is closer to users, has powerful computation potential, and well-connected to the mobile devices via a high-speed local area network. As conventional mobile cloud computing often comes with high latency, which degrades the QoS and QoE of latency-sensitive, interactive applications. The use of cloudlet is a strong need in MCC paradigm. Several applications are proposed using this architecture. The architecture proposed in [17] 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 instructions 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. Also, a two-tire (local and public clouds) networking approach has been used in [18] to enhance the QoS and scalability of mobile applications. In this work, the mobile application is modeled as a workflow of tasks, with the design goal of optimally decompose the set of tasks to execute on the mobile client and 2-tier cloud architecture.

With the increasing processing capacity of modern mobile devices, peer-assisted computing is an emerging topic that is attracting growing attention in the research community. This paradigm regards users in vicinity as an **ad hoc mobile cloud**, where neighboring mobile devices are able to utilize and share resources. Such mechanism provides an advantage of having less offload latency and

Authorized licensed use limited to: The Chinese University of Hong Kong CUHK(Shenzhen). Downloaded on May 18,2020 at 02:48:48 UTC from IEEE Xplore. Restrictions apply.

bandwidth consumption as compared to cloud computing, since mobile devices could communicate with each other via device-to-device (D2D) connections [19]. Various applications have been proposed for taking consideration of user cooperation over the mobile ad hoc networks (MANET). For example, ad-hoc streaming [20], [21], peer-to-peer media provisioning [22] and cooperative downloading [23] allow mobile users to download video collaboratively. Following the idea of distributing computational tasks to members of a MANET, previous research work [24] has explored the possibility of deploying peer-assisted multiplayer cloud gaming, especially to utilize 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. Similarly, a gaming platform is proposed in [12], which enables interactive multiplayer 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.

Users' mobility is an important influencer when running decomposed applications in cooperative manner [25]. As identified in [26], [27], and [13], researchers have realized that users' mobility pattern is influenced by their social relationships. In these works, **social closeness** between users, which is determined based on the amount of time and the frequency of communication, is identified as the best indicator of social tie, which in turn indicates the connection probability and contact duration between them. Several research works are proposed considering the social relationships between users. For instance, in [26], a mobility model is proposed to be built upon social network theory. This model puts mobile devices into groups based on the social relationships between users. These groups are then mapped to topographic spaces, with movement influenced by the strength of social tie. The evaluation shows that this model provides good approximation of real movements in terms of the distribution of contact duration, which fulfilled the design goal of such mobility model. Similarly, in [28], a social closeness method is proposed to detect clone attacks. In this work, a new metric, called community betweenness, is defined based on the social closeness between users. The authors claim that this metric changes in value when the clone attack is taken place, so that it could be used for clone attack detection.

Our proposed architecture targets **component-based game model** with game data separated from game codebase, and game logic divided into self-contained, as well as remote executable and reusable software components. In component-based game design, every object in the game (e.g., vehicles, bullets, etc.) is defined as an game entity, and every entity consists of one or more components which define its behaviour or functionality. This model provides increased system parallelism since it groups together relevant attributes and decouples the functionality which operates on them. Computation task, in this design pattern, is defined as an individual function call for certain component. As the component itself is stateless, it is the task/function parameters that go into the component which makes the task different from others. Several research works have studied the possibility of providing component-based cloud gaming services. For example, architecture proposed in [29] enables dynamic partitions of game applications so that resources of both the mobile devices and the cloud are utilized. Similarly, a cognitive platform has been designed [30] 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.

Therefore, in order to provide workload offloading with reduced data and task migration, we categorize game components into allocatable game logic components (i.e., the task for executing such components could be dynamically distributed among users, often has high computational complexity) and non-allocatable game logic components (i.e., the task for executing such component could be executed by local device only, generally have lower requirement in computation resources and manage critical user-specific data). In this component-based game model, the components could be removed or added to the entity in the runtime, thus changes the behaviour of the entity during the gameplay. Then, our progressive downloading process acts as a continuous process, yet works on demand: only the components needed for the current game progress are downloaded as a starting point for execution, while the rest is downloaded as requested (e.g., when reaching the new progress level in mobile games), with the components being added to the entities in the runtime. Both the cloudlet(s) and devices in ad hoc mobile cloud are considered to be complementary downloading sources depending on their game progresses and resources owned. Similarly, since all users in the network download game resources progressively, computation tasks could only be distributed to users (and the cloudlet(s)) who already have the associated game components downloaded for execution.

While each computation task is considered as an independent unit of computation thus could be remotely executed in parallel, the bottleneck for distributing tasks to various devices for execution is the inter-components dependencies the latency/bandwidth incurred by data transmission when sending the task over network, especially for adjacent tasks with interactive dependency (data sharing among tasks) and real-time dependencies (computation of such tasks need to be finished in timely manner, for example, tasks which respond to gun firing). Hence, to ensure the performance of distributed task processing over ad-hoc mobile network, it is inevitable for us to consider task dependencies in the task allocation process. However, in this paper, our task allocation scheme does not consider the task dependencies. Thus, for current stage, our system is more appropriate to be used in games with high computation complexity, yet low delay requirement and less real-time user interaction. One example of such game could be strategic role-playing games (SRPG, also known as tactical role-playing games), where each game player employs a number of avatars on the battlefield and the combat takes place on the battlefield without screen changes. In SRPGs,

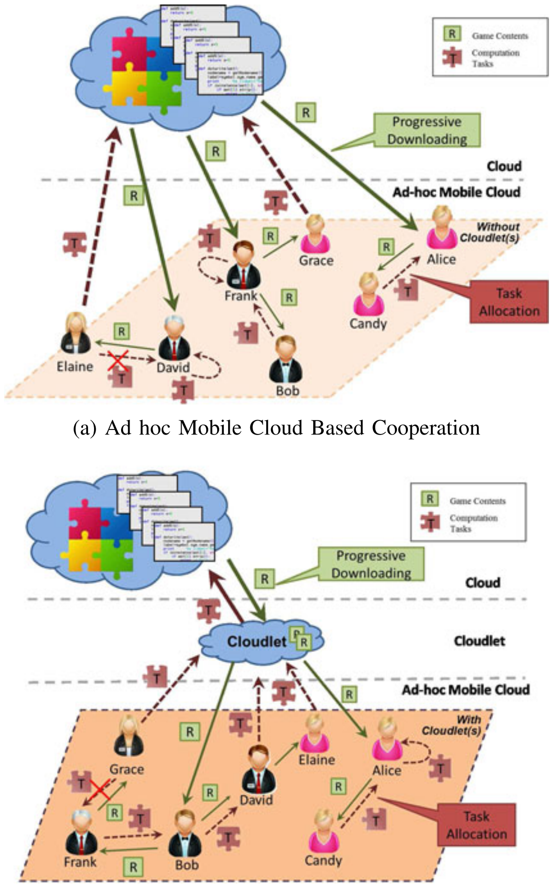
Authorized licensed use limited to: The Chinese University of Hong Kong CUHK(Shenzhen). Downloaded on May 18,2020 at 02:48:48 UTC from IEEE Xplore. Restrictions apply.

game players take turns to attack each other depending on the avatars' statistics, capabilities, and current positions. Such game has high computation complexity since 1) it incorporates strategic gameplay, i.e., tactical movement on an isometric grid and usually has complex game scene rendering; 2) it has low requirement on players' responsiveness therefore less latency-sensitive, since it does not have real-time user interactions; 3) it does not have screen changes hence less real-time rendering tasks. Therefore, it would be a good candidate for our targeting game type. For similar reasons, other strategy games such as multiplayer on-line chess would also be our targeting game type, with the benefit of enabling multiplayer gameplay. On the contrary, the feasibility to run other games that is latency-sensitive and has real-time user interaction (i.e., First Person Shooting Games, i.e., Counter-Strike) with our system is yet to be investigated at this time (without consideration of task dependencies), since delays imposed by task dependency might not be beneficial towards the gameplay experience. As the ability for such game to be decomposed and run in distributed manner remains unexplored, we consider enabling integration of such game to our system as future work, with respect of task dependency.

As previously mentioned, the proposing system could be used in **scenarios** where people need to spend an extended period of time in a crowded environment (i.e., while spending time in coffee shop or taking public transit), for which a temporary mobile ad hoc network could be formed. This concept, defined as crowd gaming, has been studied in recent researches: authors in [12] and [31] investigated the possibilities and probabilities for people playing games in public transport. They carried out measurement studies to show that the average time people spend on one-way transportation is fairly long, and most of them are willing to play multiplayer mobile games with random people in vicinity to kill time. Inspired by those studies, we believe that with the idea of crowd gaming, combining with the targeting game type, the proposing system provides a promising way for mobile gamers to paly cooperative games with minimized cellular gaming traffic generated. Such application scenario would be descripted as: a group of people is taking a subway and decide to play multiplayer online chess together using the proposing system. Then, game tasks could be distributed among themselves instead of solely offloading to the cloud. The benefits of using proposing system are twofold: 1) the device-cloud network connectivity could be intermittent when people riding fast moving underground subways. The proposing system minimizes the data transmission through the intermittent device-cloud network links thus ensures the communication and cooperativeness between gamers, while having the cloud as a backup processing unit; 2) the proposing system uses ad hoc mobile cloud to reduce cellular traffic generated while ensuring the overall system performance, which is considered a promising way to alleviate the likely to be overloaded cellular traffic, as indicated by authors in [32] and [33].

3 Architecture Overview

The basic idea of the proposed architecture is to enable nearby users who are in possession of mobile devices to play games in a cooperative manner, including progressive and collaborative game resources downloading, and cooperative task allocation. In addition to obtain services from the distant cloud, stationary cloudlet is considered as a supplementary alternative to request services from. Furthermore, instead of connecting to the cloud (over either a 3G/4G cellular connection or wireless network) and the cloudlet, nearby users are able to utilize and share resources, including processing and storage, over an ad-hoc mobile cloud. Then, two types of cooperation are provided: 1) for users who are out of the wireless range of the cloudlet, they will play games over ad-hoc mobile cloud based cooperation; 2) Along with the cooperation between peers within the ad-hoc network, users who are within the wireless range of the cloudlet will get assistance from the cloudlet.



(a) Ad hoc Mobile Cloud Based Cooperation

(b) Cooperation With The Cloudlet(s)

Fig. 1. Application scenario.

Users whom are out of the wireless range of the cloudlet connect to the cloud over cellular connection. Thus, in ad hoc mobile cloud based cooperation, two transmission schemes are being considered: 1) transmissions within the ad hoc mobile cloud via D2D connectivity; 2) transmissions between users and the cloud over cellular network. This application scenario is illustrated in Fig. 1a, where the ad-hoc mobile cloud 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

Authorized licensed use limited to: The Chinese University of Hong Kong CUHK(Shenzhen). Downloaded on May 18,2020 at 02:48:48 UTC from IEEE Xplore. Restrictions apply.

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 neighborhood ad-hoc mobile cloud 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 neighbors. Also, our system prioritizes the ad-hoc mobile cloud when offloading the computation tasks, and only offloads to the cloud when the ad-hoc mobile cloud 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.

On the contrary, users whom are within the wireless range of the cloudlet connect to the cloud through the cloudlet. Thus, in ad hoc mobile cloud based cooperation with cloudlet, three transmission schemes are being considered: 1) transmissions within the ad hoc mobile cloud via D2D connectivity; 2) transmissions between users and the cloudlet over LAN network; 3) transmissions between the cloudlet and the cloud over Wide Area Network (WAN) network. This application scenario is illustrated in Fig. 1b. In addition to the ability for users to utilize and share resources over the ad hoc mobile cloud, in this approach, the cloudlet acts as a direct neighbor to users within its wireless range. As progressively downloaded games resources from the cloud go through the cloudlet, these game resources are cached in the cloudlet. Then, the cloudlet could be considered as a great alternative to acquire the resources from. Moreover, if both the cloudlet and the ad-hoc mobile cloud do not have the required resources, users could request for services from the cloud through the cloudlet.

We focus on progressive downloading and task allocation processes since they are the determining factors for our system performance. For simplicity, we allow only unicast transmissions within the ad-hoc mobile cloud since offloading to the cloud and the cloudlet uses one-to-one transmissions. Since each transmission scheme has its own characteristics in terms of energy costs, bandwidth consumption, and interactive latency, aiming at optimizing the system performance, it is critical for the system to determine to whom to map downloading requests and computation tasks to.

4 System Modeling

In this section, we formulate both the progressive downloading and task allocation mechanisms as optimization problems, where discrete time is assumed for both mechanisms. In the formulation, users' tasks offloaded to the cloud are considered the same as offloaded to the cloudlet since 1) communication between users and the cloud is carried out by the cloudlet; 2) the cloudlet connects to the cloud over WAN network, we assume the latency in WAN network to be small and the bandwidth to be large enough, thus its associated bandwidth consumption and interactive latency could be ignored; 3) we also ignore the energy costs incurred by the cloudlet since it is a stationary device. Then, the formulations for two types of the cooperation are the same: 1) in ad hoc mobile cloud based cooperation with cloudlet, users offloads their tasks to either the cloudlet or their ad-hoc neighbors; 2) in ad hoc mobile cloud based cooperation, users offloads their tasks to either the cloud or their ad-hoc neighbors. Thus, in this section, only the formulation for ad hoc mobile cloud based cooperation is presented.

4.1 Progressive Downloading of Gaming Resources

For each user, we denote game resources downloaded from the cloud as $d_i^{3G/4G}(t)$, and game resources acquired from the ad-hoc mobile cloud as $d_i^{D2D}(t)$. Then, the overall downloading for a period of T time slots can be formulated as:

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

To achieve the maximized cooperation among users, we need to minimize the number of downloaded contents from the cloud, which can be formulated as:

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

The intention for this optimization problem is to map as much downloading requests to ad hoc mobile cloud as possible. The benefit is twofold: 1) reduce the duplicated downloading from the cloud; 2) minimize the device-cloud bandwidth consumption.

4.2 Cooperative Task Allocation for Gaming Tasks

Our task allocation process 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 neighbours. 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. In this process, both the devices reside in the network and the computation tasks play important roles. We assume that the network is constituted by a set of users (devices) ω. In general, at time t, each user $\mu_i \in \omega$ could be uniquely identified using the following attributes: 1) **Storage Availability**: k_i, which is the memory space available; 2) **CPU Availability**: z_i, which is the cpu resources available; 3) **Energy Availability**: b_i, which is current battery Level; 4) **Availability**: A_i, which is the availability to participate in the cooperation process; 5) **Energy Consumption/unit time**: e_i; At time t, user μ_i generates a computation task n_i, which is assumed to be independent in terms of execution sequence. Then, each task n_i could be uniquely identified using the following attributes: 1) **Storage Consumption**: s_i, which is the memory space required in order to compute the task; 2) **CPU Consumption**: c_i, which is the cpu resources required in order to compute the task; 3) **Task Duration/deadline**: d_i, which is the maximum time interval allowed for the task to be computed; 4) **Size of data sent**: x_i, which is the size of

Authorized licensed use limited to: The Chinese University of Hong Kong CUHK(Shenzhen). Downloaded on May 18,2020 at 02:48:48 UTC from IEEE Xplore. Restrictions apply.

TABLE 1
System Usage Parameters

Usage Parameters	Components	Device-To-Device	Device-To-Cloud
Energy Costs	Task Execution	$e_j q_{ij}$	$e^c q_i^c$
	Task Data Transmission	h_{ij}	h_i^c
	Task Results Transmission	g_{ij}	g_i^c
Bandwidth Consumption	Bandwidth	v_{ij}	v_i^c
Interactive Latency	Execution latency	q_{ij}	q_i^c
	Task Data Transmission	x_i/v_{ij}	x_i/v_i^c
	Task Results Transmission	r_i/v_{ij}	r_i/v_i^c
	Link Latency (One-way Transmission)	l_{ij}	l_i^c

the task parameters, including associated user inputs transmitted when offloading the task; 5) **Size of data received**: r_i, which is the size of the computation results received by the task owner after the task is successfully computed; 6) **Execution Time**: q_{ij}, which is the duration spent by device μ_j in computing task n_i; 7) **Bandwidth Consumption**: v_{ij}, which is the bandwidth consumed by transmitting the task over the Network; and 8) Network Link Delay: l_{ij}. To model users in the ad hoc mobile cloud, we define an N by N **neighbourhood matrix**, X, where each element represents relationship between two users. $X[i,j] = 1$ 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, $X[i,i]$ should always be 1.

4.2.1 System Resource Usage

The design goal of the task allocation process is to minimize the expected system resource usage, which can be characterized by three main parameters: the overall energy costs, bandwidth consumption, and the interactive latency. These three parameters impact differently toward the user-perceived game-play experience, thus, we consider and formulate each of them individually. Again, the ultimate goal is to map as many tasks to resources in ad hoc mobile cloud as possible in order to reduce the device-cloud bandwidth consumption.

The **energy costs** for the task allocation process include both the energy used for task processing and the energy costs incurred by data transmission across the network. To be more specific, for each task offloaded to device μ_j, we define three components in terms of energy costs, including the energy costs for executing task n_i on device μ_j, the energy costs for transmitting task parameters to the device μ_j, and the energy costs for receiving computed results from the device μ_j. Terms used to represent each component are summarized in Table 1. Combining these components together, the usage factor of energy costs for offloading task n_i to device μ_j is defined as:

$$U_{e_{ij}} = e_j(t)q_{ij}(t) + h_{ij}(t) + g_{ij}(t). \tag{3}$$

Similarly, the usage factor of energy costs for offloading task n_i to the centralized cloud c is defined as:

$$U_{e_i^c} = e^c(t)q_i^c(t) + h_i^c(t) + g_i^c(t). \tag{4}$$

Then, the overall energy costs for the task allocation process f_e could be defined as:

$$\begin{aligned} f_e = & \sum_{i=1}^{N}\sum_{j=1}^{N} F_{ij}(t)X_{ij}(t)A_j(t)U_{e_{ij}} \\ & + \sum_{i=1}^{N}\left(1 - \sum_{j=1}^{N} F_{ij}(t)X_{ij}(t)A_j(t)\right)U_{e_i^c}, \end{aligned} \tag{5}$$

where

$$F_{ij}(t) = \begin{cases} 1, & \text{If user i's task is mapped to user j} \\ 0, & \text{Otherwise.} \end{cases} \tag{6}$$

Following the same pattern, we denote v_{ij} and v_i^c as the **bandwidth consumption** for the tasks offloaded to the ad hoc mobile cloud and the cloud respectively. The usage factor of bandwidth consumption for offloading task n_i to device μ_j is defined as:

$$U_{b_{ij}} = v_{ij}(t). \tag{7}$$

Similarly, the usage factor of bandwidth consumption for offloading task n_i to the centralized cloud c is defined as:

$$U_{b_i^c} = v_i^c(t). \tag{8}$$

Then, the bandwidth consumption for the task allocation process f_b could be defined as:

$$\begin{aligned} f_b = & \sum_{i=1}^{N}\sum_{j=1}^{N} F_{ij}(t)X_{ij}(t)A_j(t)U_{b_{ij}} \\ & + \sum_{i=1}^{N}\left(1 - \sum_{j=1}^{N} F_{ij}(t)X_{ij}(t)A_j(t)\right)U_{b_i^c}. \end{aligned} \tag{9}$$

Lastly, the **interactive delay** is not just the task processing time but also includes the time it takes for data transfer across the network, which is determined by the bandwidth available in the network. Thus, we define three components for interactive latency, including the transmission latency for sending the task parameter to μ_j, the transmission latency for receiving the computed results from μ_j; as well as the execution time. The usage factor of interactive latency for offloading task n_i to device μ_j is defined as:

$$U_{l_{ij}} = x_i(t)/v_{ij}(t) + l_{ij}(t) + r_i(t)/v_{ij}(t) + l_{ij}(t) + q_{ij}(t). \tag{10}$$

Similarly, the usage factor of interactive latency for offloading task n_i to the centralized cloud c is defined as:

$$U_{l_i^c} = x_i(t)/v_i^c(t) + l_i^c(t) + r_i(t)/v_i^c(t) + l_i^c(t) + q_i^c(t). \tag{11}$$

Then, the interactive latency for task allocation f_l could be defined as:

$$\begin{aligned} f_l = & \sum_{i=1}^{N}\sum_{j=1}^{N} F_{ij}(t)X_{ij}(t)A_j(t)U_{l_{ij}} \\ & + \sum_{i=1}^{N}\left(1 - \sum_{j=1}^{N} F_{ij}(t)X_{ij}(t)A_j(t)\right)U_{l_i^c}. \end{aligned} \tag{12}$$

Targeting at the minimized system resource usage, we want the energy costs, bandwidth consumption, and

Authorized licensed use limited to: The Chinese University of Hong Kong CUHK(Shenzhen). Downloaded on May 18,2020 at 02:48:48 UTC from IEEE Xplore. Restrictions apply.

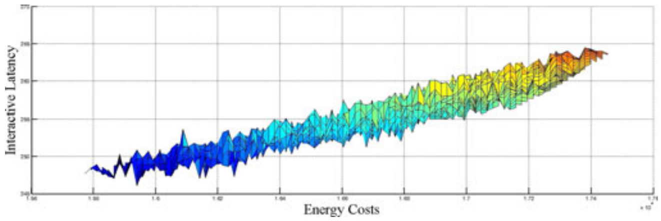


Fig. 2. Interactive latency versus energy costs.

interactive latency to be minimized. Then, the system resource usage function could be considered as multi-objective optimization which involves minimizing all three usage parameters

$$\text{Minimize}: (f_e, f_b, f_l) \tag{13}$$

$$\text{Subject to}: \sum_{i=1}^{N} F_{ij}(t) * X_{ij} * A_j(t) * s_j(t) \le k_j(t), \forall j, \tag{14a}$$

$$\sum_{i=1}^{N} F_{ij}(t) * X_{ij} * A_j(t) * c_j(t) \le z_j(t), \forall j, \tag{14b}$$

$$\sum_{j=1}^{N} F_{ij}(t) * X_{ij} * A_j(t) \le 1, \forall i, \tag{14c}$$

$$F_{ij}(t) \in \{0, 1\}. \tag{14d}$$

Before we introduce the constraints in the formulation, it is important for us to study the relationships between the objective functions in the multi-objective optimization problem. After evaluating the value of each objective function using a fixed set of parameters (with parameters randomly generated) for users and tasks, as well as the fixed (randomly generated) network topology, we find that three objective functions are conflicting. They depend on and restrict each other mutually. To be more specific, as shown in Figs. 2 and 3, the interactive latency increases as energy costs increases, yet decreases as the bandwidth increases. In this case, instead of a single solution that simultaneously optimizes each objective, there exists a set of Pareto optimal solutions (Pareto front), which is shown in Fig. 4. Since all Pareto optimal solutions are considered equally good, trade-offs need to be considered in order to solve such multi-objective optimization problem.

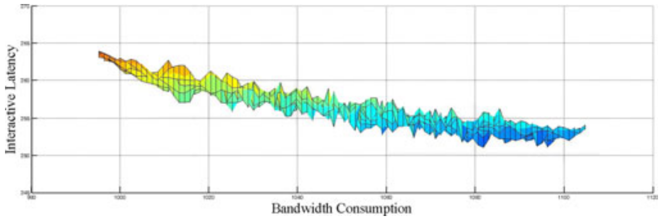


Fig. 3. Interactive latency versus bandwidth consumption.

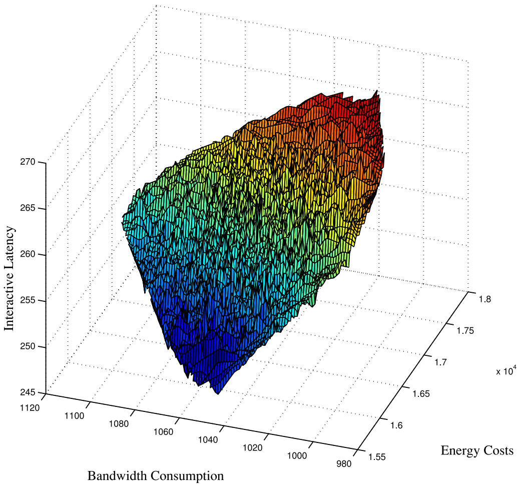


Fig. 4. Pareto front: a set of pareto optimal solutions to the system resource usage problem.

To qualify the trade-offs, we incorporate a set of weight factors: $\{\alpha, \beta, \delta\}$, where $\alpha + \beta + \delta = 1$, to reflect the relative importance of the energy costs, bandwidth consumption, and interactive latency respectively. These weight factors are pre-set by the system to default numbers (could be future adjusted by users to suit their own needs, with the constraint remaining $\alpha + \beta + \delta = 1$), which are adaptively determined according to the computation complexity of the game application, game data volume, as well as its interactive frequency (Computation, Data, and Interaction, where $Computation \in [0, 1]$, $Data \in [0, 1]$, and $Interaction \in [0, 1]$). Indeed, some games are interaction-intensive, while others being either data-intensive or computation-intensive. For instance, as shown in Table 2, games like Chess has high value on computation complexity, less of the interaction frequency and very little on game data volume; the famous Massive Multi-player On-line Game (MMOG) called Diablo3 has higher value on game data volume than the interaction and the computation complexity (note that the Computation, Data, and Interaction value given in the examples are user-perceived, it does not have scientific meanings). Thus, for interaction-intensive game applications, it is reasonable to set the importance factor of the interactive latency δ higher than the energy costs and the bandwidth consumption; on the contrary, for computation-intensive game applications, the importance factor of the energy costs α should be higher among the three of the factors; similarly, for data-intensive game applications, we need to set the importance factor of the bandwidth consumption β higher than the others as it needs more data exchanges. Then, the system resource usage could be modeled as a single objective optimization problem, shown as below:

$$\begin{aligned}\text{Minimize}: &\sum_{i=1}^{N}\sum_{j=1}^{N} F_{ij}(t)X_{ij}(t)A_j(t)\\ &\left(\alpha U_{e_{ij}} + \beta U_{b_{ij}} + \delta U_{l_{ij}}\right)\\ &+ \sum_{i=1}^{N}\left(1 - \sum_{j=1}^{N} F_{ij}(t)X_{ij}(t)A_j(t)\right)\\ &\left(\alpha U_{e_i^c} + \beta U_{b_i^c} + \delta U_{l_i^c}\right)\\ \text{Subject to}: &\ (14a),(14b),(14c),(14d).\end{aligned} \tag{15}$$

TABLE 2
Examples of Game Classification

Game Application	Computation	Data	Interaction	Classification
Chess	0.9	0.1	0.4	Computation-Intensive
Diablo3	0.5	0.9	0.7	Data-Intensive
LoL	0.4	0.6	0.9	Interaction-Intensive

Authorized licensed use limited to: The Chinese University of Hong Kong CUHK(Shenzhen). Downloaded on May 18,2020 at 02:48:48 UTC from IEEE Xplore. Restrictions apply.

```
$U$ : Set of neighbours who has the required game piece
$P$ := Local game progress speed
$D := \emptyset$
while $U \neq \emptyset$ do
    choose $u \in U : T_u$ being minimal and less than $P$;
    if $u \neq Null$ then
        $G \leftarrow downloadFromNeighbour(u)$
    else
        $G \leftarrow DownloadFromCloud()$
    end if
    $D := D \cup \{G\}$;
end while
```

Fig. 5. Algorithm for progressive downloading: greedy local.

Four constraints are defined in the formulation. The first (14a) and the second (14b) 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 (14c) is to assign no more than one task to each user, and the last (14d) indicates that for $F_{ij}(t)$ we only accept solutions as 0 or 1 as defined earlier.

4.2.2 Mobile Social Closeness

In the previous section, we introduced a model which maps each computation task to the candidate with minimal system resource usage. However, tasks offloaded to neighbors may not be successfully computed and returned before the task deadline due to the mobility of users. These tasks, regarded as failed, negatively impacts the Quality of Service since they need to be re-allocated. To improve the QoS, we consider the probability of failure, R_{ij}, for the process of offloading task n_i to device μ_j , which is proposed to measure the closeness between the users: high degree in closeness, low probability of failure. Then, while constraints are remained the same, the task offloading process considering the probability of failure could be modeled as:

$$\begin{aligned}\text{Minimize}: \ & \sum_{i=1}^{N}\sum_{j=1}^{N} F_{ij}(t)X_{ij}(t)R_{ij}(t)A_j(t) \\ & \left(\alpha U_{e_{ij}} + \beta U_{b_{ij}} + \delta U_{l_{ij}}\right) \\ & + \sum_{i=1}^{N}\left(1 - \sum_{j=1}^{N} F_{ij}(t)X_{ij}(t)A_j(t)\right) \\ & \left(\alpha U_{e_i^c} + \beta U_{b_i^c} + \delta U_{l_i^c}\right) \\ \text{Subject to}: \ & (14a),(14b),(14c),(14d).\end{aligned} \tag{16}$$

5 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 algorithm 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. Again, similar to the system formulation, only algorithms used for ad hoc mobile cloud based cooperation is presented in this section.

```
$D := \emptyset$
$W$ := Set of weighted users
$K$ := Set of user being center to each cluster
$K \leftarrow KMeanClustering(W)$;
for $i \leftarrow 0$ to $N$ do
    if $W_i \in K$ then
        $G \leftarrow downloadFromCloud(G)$;
    else
        $G \leftarrow downloadFromNeighbour()$;
    end if
    $D := D \cup \{G\}$;
end for
```

Fig. 6. Algorithm for progressive downloading: weighted k-mean.

5.1 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 neighbor with the minimal transmission speed. This algorithm is summarized in Fig. 5, where a user μ_i needs a game resource that is owned by a user μ_i, 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, $P_i(t)$, and decide whether to receive the content through the server or its neighbor by comparing $T_{ij}(t)$ and $P_i(t)$: if $T_{ij}(t) < P_i(t)$, the user should be receiving the content through its neighbor; otherwise, the content is downloaded from the cloud.

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 μ_i is associated with a weight, P_i, which is the current game progress for μ_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 μ_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 Fig. 6.

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 mobile cloud, 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.

Authorized licensed use limited to: The Chinese University of Hong Kong CUHK(Shenzhen). Downloaded on May 18,2020 at 02:48:48 UTC from IEEE Xplore. Restrictions apply.

```
$N :=$ Set of Tasks
$M := \emptyset$
for $i \leftarrow 0$ to $Size(N)$ do
    $choose\{u \in neighbourOf(initiator(N_i))\}$;
    if $u \neq Null$ then
        $M := M \cup \{N_i, u\}$;
        $z_u := z_u - c_{N_i}$;
        $k_u := k_u - s_{N_i}$;
    else
        $OffloadToCloud(N_i)$
    end if
end for
```

Fig. 7. Algorithm frame for task allocation.

5.2 Cooperative Task Allocation for Ad-Hoc Mobile Gaming

The basic structure of the algorithm for this module is presented in Fig. 7. Assuming there is a mapping M, which represents the task-resource mapping within the ad-hoc mobile cloud, and we have to specify a set of tasks initiated by users within the ad-hoc mobile cloud and decide to which neighbor the task is allocated so that the system resource usage could be minimized. We use several different strategies for choosing the neighbor to offload the task, where they all have to meet one constraint, which is for the chosen neighbor to have sufficient computation potential for the task. For example, if user μ_i has a task n_i and wants to map to its neighbor μ_j, then user μ_j must have $k_j \geq s_{n_i}$ and $z_j \geq c_{n_i}$. If no valid neighbor is in the ad hoc mobile cloud, the task is then offloaded to the cloud.

- *Random mapping*: Task requests are mapped to randomly chosen neighbors who have enough processing and storage capabilities.
- *Greedy local*: Each potential task-resource mapping has its corresponding system resource usage. We use $Cost_{ij}$ to refer to the system resource usage for offloading task from user μ_i to μ_j. In this strategy, each task is assigned to the available neighbor with minimum $Cost_{i,neighbor}$, thus, the globally optimal solution can be achieved by making locally optimal choices.
- *Greedy Heuristic*: In this strategy, each task is assigned to user μ_j if it has the minimum additional $Cost_{i,j}$ based on its current mapping.

The computational and storage complexity for the above algorithms are listed in Table 3. In our analysis, we consider only the complexity for the neighbor selection process since steps for maintenance are common for all algorithms.

6 EVALUATIONS

System performance is evaluated by using a Java simulator, which assigns random attributes to the users, the computation tasks, and the offloading process. 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 mobile cloud's processing capability and hence will have to be offloaded to the cloud. Details of how each parameter is generated are listed in Table 4. 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.

TABLE 3
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)

TABLE 4
Setup Parameters

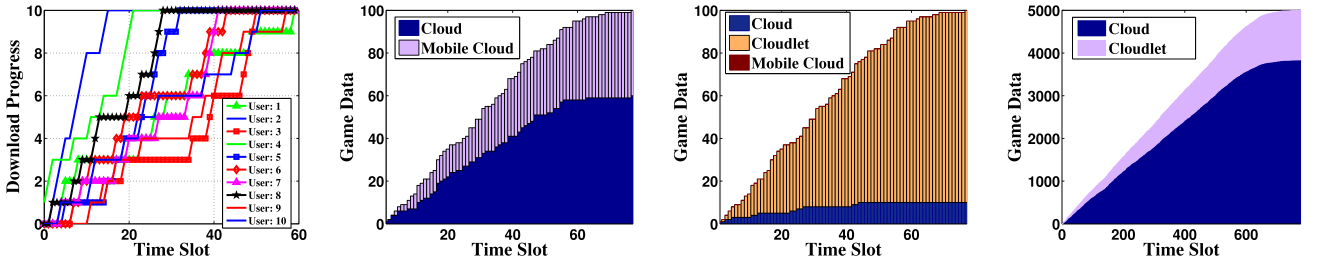
Target	Parameter	Minimum	Maximum
Base Case	Device(s), Cloud, and Cloudlet(s)	50	350
Task(s) Factors	Storage Consumption	0.9	1.05
	CPU Consumption	0.9	1.05
	Task Duration (Deadline)	0.5	0.5
	Amount Of Data Sent	5	5
	Bandwidth Consumption	1	1
	Amount Of Data Received	5	5
User(s) / Device(s) Factors	Storage Availability	1	1
	CPU Availability	1	1
	Energy Availability	0.5	0.5
	Energy Consumption (/ unit time)	0.5	0.5
	Game Progress Speed	1	10
Cloudlet(s) Factors	Storage Availability	20	20
	CPU Availability	20	20
	Energy Availability	1	1
	Energy Consumption (/unit time)	0.1	0.1
Local Level Offloading	Short On-Device Execution Time	0.5	0.5
	Long On-Device Execution Time	5	5
	Device-Device Data Transmission Costs	10	10
	Device-Cloudlet Data Transmission Costs	20	20
	Link Delay	1	2
Device-Cloud Offloading	Task Execution Time	0.1	0.1
	Device-Cloud Data Transmission Costs	30	30
	Link Delay	10	20
Network Topology	Density	1	10
	Skewness	1	10
	Task Density	1	10
Weight Factors	Energy Costs	0	1
	Bandwidth Consumption	0	1
	Interactive Latency	0	1

6.1 Randomly Generated Network Topology

6.1.1 Progressive Downloading

Each user downloads a game resource depending on their game progress speed, which is the time slots interval between each download. An example of the downloading

Authorized licensed use limited to: The Chinese University of Hong Kong CUHK(Shenzhen). Downloaded on May 18,2020 at 02:48:48 UTC from IEEE Xplore. Restrictions apply.



(a) Download Progress (b) Download Without Cloudlet(s) (c) Download With Cloudlet(s) (d) Download With Trace-driven Network

Fig. 8. Progressive download.

progress for ten users is shown in Fig. 8a. In this example, game data are segmented into ten pieces and the downloading process is implemented using Greedy Local. The experimental results in Fig. 8a show that users with faster game progression speed finishes downloading before others.

For ad hoc mobile cloud based cooperation, game resources can be acquired from either ad-hoc neighbors or the cloud. Thus, we differentiate the downloaded game pieces via different network links. Fig. 8b shows the total downloaded game data for all users. Game pieces downloaded directly from the cloud (via cellular links) are colored in blue whereas pieces acquired from the ad hoc mobile cloud (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 mobile cloud. Thus, total downloading traffic via cellular link is significantly reduced.

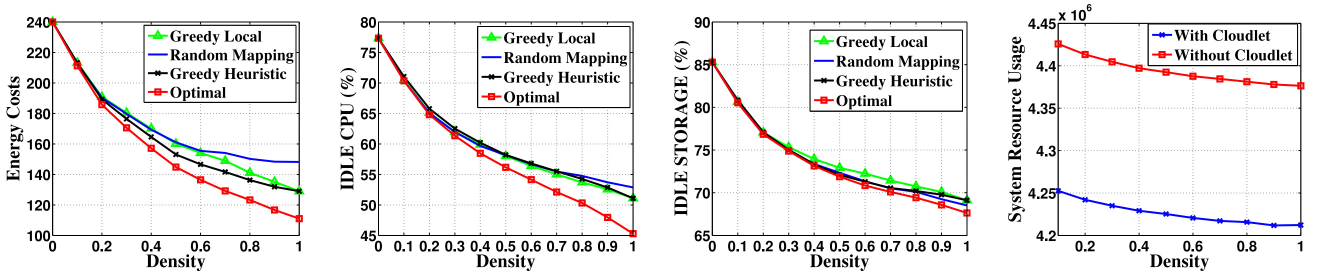
The total downloading for ad hoc mobile cloud based cooperation with the assistance from the cloudlet(s) is shown in Fig. 8c. Again game pieces downloaded directly from the cloud are colored in blue whereas pieces acquired form the ad hoc mobile cloud are colored in red. Here we have an additional downloading source, which is the cloudlet(s), and game pieces downloaded from the cloudlet(s) are colored in yellow. From the graph, we can see that as the game progresses, most of the game resources are acquired from the cloudlet(s). This is because that all game pieces downloaded from the cloud are cached in the cloudlet(s), so that most of the users could acquire game pieces from the cloudlet(s) instead of their ad-hoc neighbors. This indicates that by adding the cloudlet(s), the downloading performance is improved significantly.

Actually, bring along the cloudlet into our architecture has multiple advantages. For example, without the assistant from the cloudlet, users may only acquire game resources from either their direct neighbors or the cloud. It ignores the possible content sharing between multi-hop neighbors, which limits the efficiency of the content sharing. Alternatively, if the downloaded resources are cached in the cloudlet, content sharing between multi-hop neighbors could be carried out by the cloudlet, without multi-hop data transmission. Also, as the ad hoc mobile cloud is actually an opportunistic network, the probability for which content sharing in ad hoc mobile cloud based cooperation occurs depends on the time and users' physical locations. On the contrary, the cloudlet is considered to be a stable device as compared to mobile devices. It provides potential content sharing for users who have the same routine, but not reside in the same ad hoc mobile cloud.

6.1.2 Task Allocation for Ad Hoc Mobile Cloud Based Cooperation

We first evaluated the performance for the ad hoc mobile cloud based cooperation under different randomly generated network environments. Our Proposed algorithms are used to implement the task allocation process, including the solution to the optimization problem formulation (optimal solutions), which is produced using optimization solver called Gurobi, 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. Also, for the simulation results shown below, we does not include the task re-assignment and assume that each task offloaded could be successfully computed and returned before its associated deadline.

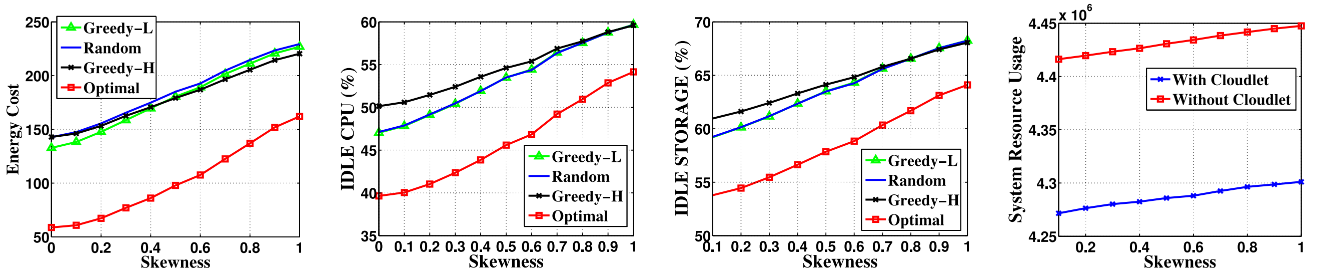
Density. Fig. 9a shows the overall energy costs incurred by different allocation algorithms under different network environments. The network topology is generated in response to different degrees of network density, where a topology with higher density indicates that there is more



(a) Energy Costs (b) CPU Utilization (c) Storage Utilization (d) System Resource Usage With and Without Cloudlet

Fig. 9. System performance for various density.

Authorized licensed use limited to: The Chinese University of Hong Kong CUHK(Shenzhen). Downloaded on May 18,2020 at 02:48:48 UTC from IEEE Xplore. Restrictions apply.



(a) Energy Costs

(b) CPU Utilization

(c) Storage Utilization

(d) System Resource Usage With and Without Cloudlet

Fig. 10. System performance for various skewness (in the legend: greedy-l stands for greedy local; random stands for random mapping; greedy-h stands for greedy heuristic; optimal stands for optimal.

D2D connectivity between users than a topology with lower density. These experimental results show that the energy costs decrease as the network density increases. This is because a higher degree in density means users have more connected neighbors, so that more tasks can be offloaded to the ad hoc mobile cloud, and the costs to offload tasks to the ad hoc mobile cloud are often smaller than the alternative. Figs. 9b and 9c 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. 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. 10a, 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 neighbors while only a small number of users have most of the connections. Sometimes users with few neighbors 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 energy costs are minimized. Also, our system has better ad hoc mobile cloud utilization when the skewness is minimal, as showed in Figs. 10b and 10c. 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.

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 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 and balanced. 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 3.

6.1.3 *Task Allocation for Ad Hoc Mobile Cloud Based Cooperation with the Assistance from the Cloudlet(s)*

We then evaluated the performance for ad hoc mobile cloud based cooperation with the assistance from the cloudlet. The optimal solutions obtained for both types of network topology, as shown in Figs. 9d and 10d, are compared side by side. From the figures, we can see that with the assistance from the cloudlet, the system resource usage decreases as the network density increases, yet increases as the network skewness increases, which is the same as in the ad hoc mobile cloud based cooperation. However, the system resource usage with the cloudlet is much lower than without the cloudlet, which indicates that by adding the cloudlet into our architecture, the system performance is improved.

6.2 Real-World Trace-Based Network Topology

To evaluate the effect of user's mobility on our system performance, we simulated the the ad hoc mobile cloud based cooperation using a trace-driven network topology generator called ONE. Map based movement patterns is used, where users are located inside several street blocks initially and their movement patterns are according to the map of each block.

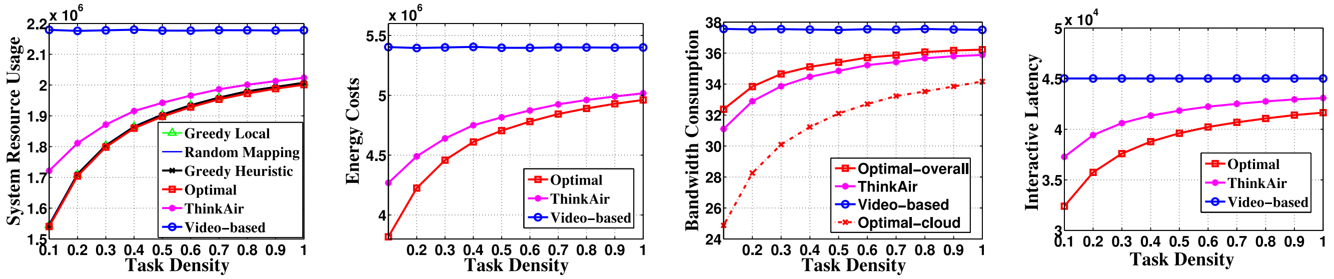
6.2.1 *Progressive Downloading*

Experimental Result for progressive downloading under map-based movement topology is shown in Fig. 8d respectively. From these figures, we can see that as a game progresses, more and more game data are acquired from the ad hoc mobile cloud and total traffic via the cellular network is reduced.

6.2.2 *Task Allocation without Closeness*

In this section, the effect of different on-device task duration on the system performance is evaluated. We consider

Authorized licensed use limited to: The Chinese University of Hong Kong CUHK(Shenzhen). Downloaded on May 18,2020 at 02:48:48 UTC from IEEE Xplore. Restrictions apply.



(a) System Resource Usage (b) Energy Costs (c) Bandwidth Consumption (d) Interactive latency

Fig. 11. Task allocation with map-based movement: short on-device task duration.

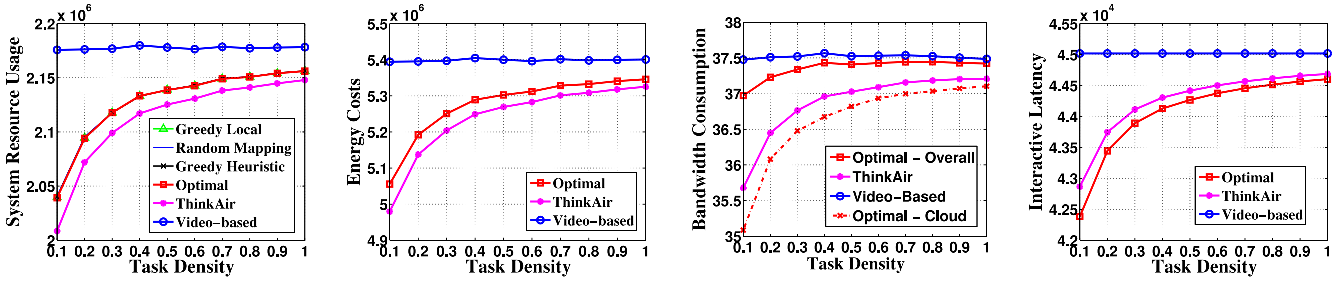
task-reassignment, where each task that is allocated to adhoc neighbors yet failed to return computation results before the task deadline are re-assigned. We then simulate the task allocation process with both short on-device task duration and long on-device task duration for different task density (the probability of users having tasks to offload at each time slot). The results for system resource usage, energy costs, bandwidth consumption, interactive latency per task are compared with 1) the conventional cloud based architecture, where all computation tasks are offloaded to the cloud; 2) the mobile code offloading based architecture (ThinkAir) proposed in [34], where computation tasks are offloaded to the cloud according to its task offloading policy: computation tasks are offloaded to the cloud only if both the task execution time and the energy consumed to carry out the computation will improve (reduce).

The experimental results for short on-device task duration are shown in Figs. 11a, 11b, 11c, and 11d, respectively. From the figures we can see that the system resource usage, the energy costs, the bandwidth consumption, and the interactive latency incurred by our proposed algorithms increase as the task density increase yet at a decreasing rate. This is because: 1) at a higher task density, more task in percentage are offloaded to the cloud thus no task re-assignment is taken place; 2) at a lower task density, more task in percentage are offloaded to the ad hoc mobile cloud, thus will have higher task re-assignment probability (the probability for which the communication link between neighbors are broken before the computation results could be successfully returned) and therefore higher system resource usage, the energy costs, the bandwidth consumption, and the interactive latency. Also, we observe that our task allocation mechanism has lower system resource usage, energy costs and interactively latency than both the conventional cloud based architecture and the ThinkAir. In our mechanism, we consider two types of bandwidth consumption, both device-device (for tasks offloaded to the ad hoc mobile cloud) and device-cloud (for tasks offloaded to the cloud). From the result, we see that our mechanism has higher overall bandwidth consumption than the ThinkAir, yet lower device-cloud bandwidth consumption. This is because that in our mechanism, tasks are distributed within the ad hoc mobile cloud, which eaten up the device-device network bandwidth. However, as the available device-device bandwidth are considered to be large and much cheaper than as in the cellular networks, this result is acceptable.

Then, the experimental results for long on-device task duration are shown in Figs. 12a, 12b, 12c, and 12d. From the figures, we can see that our mechanism has 1) lower interactive latency than both the conventional cloud based architecture and the ThinkAir; 2) higher system resource usage and energy costs. This is because in this scenario, the ThinkAir offload more tasks to the cloud, whereas our mechanism prioritizes the ad hoc mobile cloud in the offloading process, therefore higher energy costs are incurred by our mechanism with longer on-device task duration. Also, although the overall bandwidth consumption is higher than the ThinkAir, our mechanism has lower device-cloud bandwidth consumption due to ad hoc mobile cloud collaboration. As the design goal of our architecture is to utilize the resources of the ad hoc mobile cloud and the cloudlet(s) to achieve reduced data transmission between end users and the distant cloud, it is safe for as to say that the design goal is met. Also, from the results shown above, we observe that our proposed scheme performs better in the scenario with short on-device task duration.

6.2.3 *Task Allocation with Closeness*

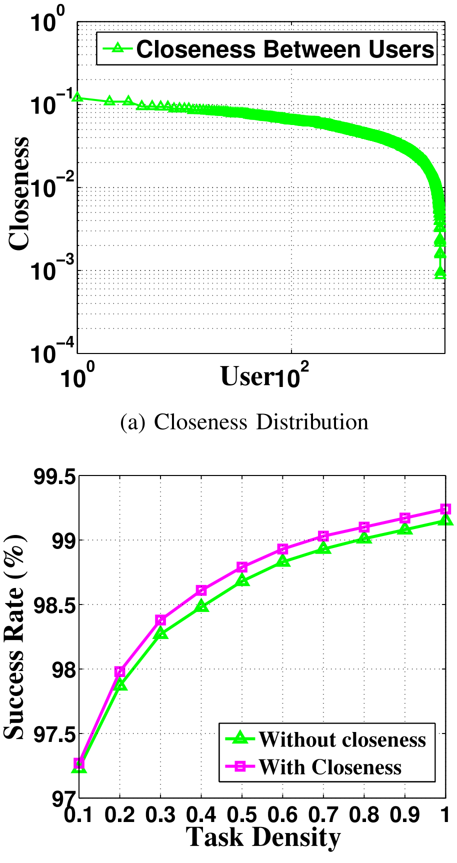
To show the effect of closeness in the task offloading process, the success rate for the task allocation process with and without the consideration of the closeness are evaluated. Here we define the closeness between users as the total



(a) System Resource Usage (b) Energy Costs (c) Bandwidth Consumption (d) Interactive Latency

Fig. 12. Task allocation with map-based movement: long on-device task duration.

Authorized licensed use limited to: The Chinese University of Hong Kong CUHK(Shenzhen). Downloaded on May 18,2020 at 02:48:48 UTC from IEEE Xplore. Restrictions apply.



(a) Closeness Distribution

(b) Success Rate with/without Closeness

Fig. 13. Evaluation for mobile social closeness with map-based movement and system overhead.

contact time/simulation time and the closeness distribution for map-based movement are shown in Fig. 13a. In the simulation, we compare the solutions to the problem formulation introduced in (15) and (16) for task allocation with and without the closeness respectively. The probability of failure R_{ij} in (16) is defined as a number inversely proportional to the closeness between user μ_i and μ_j. Experimental results are shown in Figs. 13a and 13b, respectively. In the evaluation, we regard each task that is assigned to ad-hoc neighbors, yet failed to return the computation results within the task duration as failed. From the results, we observe that by considering closeness in the task offloading process, the success rate is improved, which indicates that by considering the closeness the system performance will be improved.

6.3 Overhead and Benefit

Although the proposing tiered architecture are shown to be beneficial in terms of reducing device-cloud cellular data transmission, enabling cloudlet and ad hoc mobile cloud offloading brings additional system usage, i.e., additional device-to-device bandwidth and energy consumed for data transmission. Then, in order to give a comprehensive analysis of the system performance, it is important to show that the benefit realized by users outweigh this additional overhead. Fig. 14 shows the overall system resource usage per unit task measured in the cooperative task allocation process. The experimental results show that the usage/task increases as the task density increases yet at a decreasing rate. To further understand the underlying reason for such phenomenon, we measure and show the system resource usage for the device-device and device-cloud transmission separately, as well as the task in percentage that is offloaded to the cloud and the ad hoc mobile cloud. The results are also shown in Figs. 14 and 15, respectively. From Fig. 14, we see that the usage/task for device-cloud transmission increases at a decreasing rate, while the usage/task for device-device transmission decreases at a decreasing rate. This means that as task density increases, the tasks in percentage being offloaded to the cloud also increases and the tasks in percentage being offloaded to the ad hoc mobile cloud decreases, which indicates that the resources of ad hoc mobile cloud are being efficiently utilized (This conclusion is also demonstrated in Fig. 14. However, the increasing/decreasing system resource usage for device-cloud/device-device transmission converges, which indicates that the resources in ad hoc mobile cloud are becoming fully utilized as the task density increases. The benefit of the system is the notably reduced system resource usage incurred by the device-cloud transmission as compared to the conventional cloud gaming, while the system overhead being the additional usage incurred by the device-device transmission. Considering the system has a decreasing device-device usage as the task density increases, and the usage incurred by the device-device transmission is relatively small, along with the fact that the device-to-device transmission is often considered being fast and cheap [19], it is reasonable for us to conclude that this overhead is the acceptable performance tradeoff.

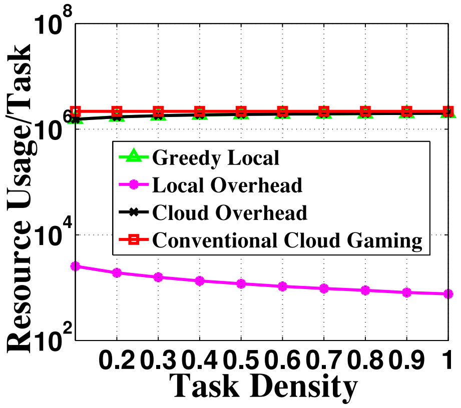


Fig. 14. Success rate with/without closeness for map based movement.

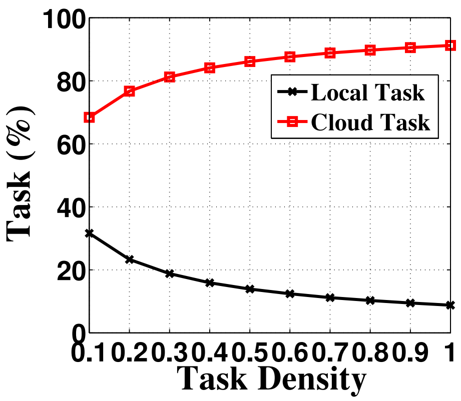


Fig. 15. Success rate with/without closeness for map based movement.

Authorized licensed use limited to: The Chinese University of Hong Kong CUHK(Shenzhen). Downloaded on May 18,2020 at 02:48:48 UTC from IEEE Xplore. Restrictions apply.

7 Conclusion

In this paper, we have proposed an ad hoc mobile cloud-cloudlet-cloud based gaming architecture, which considers both progressive and collaborative downloading of game resources as well as cooperative 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 system resource usage with short on-device task duration as 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 system resource usage. In a word, our proposed scheme minimizes the data transmissions between the cloud and the users, and optimizes the local cooperations between the users in a best effort manner. The scalability of this system is acceptable, since both progressive downloading and cooperative task processing are decentralized decision-making processes. Efficient and decentralized service discovery, device discovery, and membership management mechanisms should be carefully designed to ensure the scalability of the system, which will be considered as future work.

Acknowledgments

This work was supported by a University of British Columbia Four Year Doctoral Fellowship, the Natural Sciences and Engineering Research Council under Grant STPGP 447524, and the National Natural Science Foundation of China under Grant No. 61271182.

References

[1] M. R. Rahimi, J. Ren, C. Liu, A. V. Vasilakos, and N. Venkatasubramanian, “Mobile cloud computing: A survey, state of art and future directions,” *Mobile Netw. Appl.*, vol. 19, no. 2, pp. 133–143, 2014.

[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. CPSCom*, Beijing, China, Aug. 2013, pp. 769–776.

[3] C. Huang, C. Hsu, Y. Chang and K. Chen, “GamingAnywhere: An open cloud gaming system,” in *Proc. ACM Multimedia Syst. Conf.*, Oslo, Norway, Feb./Mar. 2013, pp. 36–47.

[4] X. Wang, A. V. Vasilakos, M. Chen, Y. Liu, and T. T. Kwon, “A survey of green mobile networks: Opportunities and challenges,” *Mobile Netw. Appl.*, vol. 17, no. 1, pp. 4–20, 2012.

[5] Z. Feng, Y. Zhu, Q. Zhang, L. M. Ni, and A. V. Vasilakos, “TRAC: Truthful auction for Location-aware collaborative sensing in mobile crowdsourcing,” in *Proc. IEEE INFOCOM*, Toroto, Canada, 2014, pp. 1231–1239.

[6] L. Zhou, N. Xiong, L. Shu, A. V. Vasilakos, S. Yeo, “Context-aware middleware for multimedia services in heterogeneous networks,” *IEEE Intell. Syst.*, vol. 25, no. 2, pp. 40–47, Mar./Apr. 2010.

[7] P. P. Demestichas, V. G. Stavroulaki, L. I. Papadopoulou, A. V. Vasilakos, and M. E. Theologou, “Service configuration and traffic distribution in composite radio environments,” *IEEE Trans. Syst., Man, Cybern. C*, vol. 34, no. 1, pp. 69–81, Feb. 2004.

[8] F. Liu, P. Shu, H. Jin, L. Ding, J. Yu, D. Niu, and B. Li, “Gearing Resource-poor mobile devices with powerful clouds: Architectures, challenges, and applications,” *IEEE Wireless Commun.*, vol. 20, no. 3, pp. 14–22, Jun. 2013.

[9] G. Wei, A. V. Vasilakos, Y. Zheng, and N. Xiong, “A Game-theoretic method of fair resource allocation for cloud computing services,” *J. Supercomput.*, vol. 54, no. 2, pp. 252–269, 2010.

[10] W. Fang, Y. Li, H. Zhang, N. Xiong, J. Lai, and A. V.Vasilakos, “On the Throughput-energy tradeoff for data transmission between cloud and mobile devices,” *Inf. Sci.*, vol. 283, no. 1, pp. 79–93, Nov. 2014.

[11] Y. Li and W. Wang, “The unheralded power of cloudlet computing in the vicinity of mobile devices,” in *Proc. IEEE Globecom*, Atlanta, GA, USA, Dec. 2013, pp. 4994–4999.

[12] S. Pushp, C. Liu, F. Liu, and J. Song, “Multi-player gaming in public transport crowd: Opportunities and challenges,” in *Proc. IEEE World Forum Internet Things*, Seoul, Korea, Mar. 2014, pp. 331–336.

[13] M. Conti, S. Giordano, M. May, and A. Passarella, “From opportunistic networks to opportunistic computing,” *IEEE Commun. Mag.*, vol. 48, no. 9, pp. 126–139, Sep. 2010.

[14] M. Satyanarayanan, P. Bahl, R. Caceres, and N. Davies, “The case for VM-based cloudlets in mobile computing,” *IEEE Pervasive Comput.*, vol. 8, no. 4, pp. 14–23, Oct.–Dec. 2009

[15] Y. Jararweh, L. Tawalbeh, F. Ababneh, and A. Khreishah, “Scalable cloudlet-based mobile computing model,” *Procedia Comput. Sci.*, vol. 34, pp. 434–441, 2014

[16] M. Satyanarayanan, P. Bahl, R. Caceres, and N. Davies, “The case for VM-based cloudlets in mobile computing,” *IEEE Pervasive Comput.*, vol. 8, no. 4, pp. 14–23, Oct.–Dec. 2009

[17] D. Meilander, F. Glinka, S. Gorlatch, L. Lin, W. Zhang and X. Liao, “Bringing mobile online games to clouds,” in *Proc. IEEE INFOCOM Workshop Mobile Cloud Comput.*, Toronto, ON, Canada, Apr. 2014, pp. 340–345.

[18] M. R. Rahimi, N. Venkatasubramanian, S. Mehrotra, and A. V. Vasilakos, “MAPCloud: Mobile applications on an elastic and scalable 2-tier cloud architecture,” in *Proc. IEEE 5th Int. Conf. Utility Cloud Comput.*, Chicago, IL, USA, Nov. 2012, pp. 83–90.

[19] Y. Li, L. Sun, and W. Wang, “Exploring device-to-device communication for mobile cloud computing,” in *Proc. IEEE Int. Conf. Commun.*, Sydney, Australia, Jun. 2014, pp. 2239–2244.

[20] K. Lorenzo, L. Anh and C. Blerim, “MicroCast: Cooperative video streaming on smartphones,” in *Proc. 10th Annu. Int. Conf. Mobile Syst., Appl. Services*, Low Wood Bay, Lake District, United Kingdom, Jun. 2012, pp. 57–70.

[21] Z. Shen, J. Luo, R. Zimmemann, and A. V. Vasilakos, “Peer-to-peer media streaming: Insights and new developments,” *Proc. IEEE*, vol. 99, no. 12, pp. 2089–2109, Dec. 2011.

[22] L. Zhou, Y. Zhang, K, Song, W. Jing, and A. V. Vasilakos, “Distributed media services in P2P-based vehicular networks,” *IEEE Trans. Veh. Technol.*, vol. 60, no. 2, pp. 692–703, Feb. 2011.

[23] J. Wang, S. Wang, Y. Sun, W. Lu, and D. Wu, “An incentive mechanism for cooperative downloading method in VANET,” in *Proc. IEEE Int. Conf. Veh. Electron. Safety*, Dongguan, China, Jul. 2013, pp. 125–130.

[24] W. Cai, V. Leung, and L. Hu, “A cloudlet-assisted multiplayer cloud gaming system,” *ACM Mobile Netw. Appl.*, vol. 19, no. 2, pp. 144–152, Apr. 2014.

[25] M. R. Rahimi, N. Venkatasubramanian, and A. V. Vasilakos, “MuSIC: Mobility-aware optimal service allocation in mobile cloud computing,” in *Proc. IEEE 6th Int. Conf. Cloud Comput.*, Santa Clara, CA, USA, 2013, pp. 75–82.

[26] M. Musolesi and C. Mascolo, “Designing mobility models based on social network theory,” *ACM SIGMOBILE Mobile Comput. Commun. Rev.*, vol. 11, no. 3, pp. 59–70, Jul. 2007.

[27] S. Phithakkitnukoon and R. Dantu, “Mobile social closeness and communication patterns,” in *Proc. IEEE 7th Consum. Commun. Netw. Conf.*, Las Vegas, NV, USA, Jan. 2010, pp. 1–5.

[28] Y. Ren, Y. Chen, and M. C. Chuah, “Social closeness based clone attack detection for mobile healthcare system,” in *MASS'2012*, Las Vegas, NV, USA, Oct. 2012.

[29] W. Cai and V. Leung, “Decomposed cloud games: Design principles and challenges,” in *Proc. IEEE Int. Conf. Multimedia Expo Workshop*, Chengdu, China, Jul. 2014, pp. 1–4.

[30] W. Cai, C. Zhou V. Leung, and M. Chen, “A cognitive platform for mobile cloud gaming,” in *Proc. IEEE 5th Int. Conf. Cloud Comput. Technol. Sci.*, Bristol, U.K., Dec. 2013, pp. 72–79.

[31] N. Zhang, Y. Lee, M. Radhakrishman, and R. K. Balan, “GameOn: p2p gaming on public transport,” in *Proc. 13th Annu. Int. Conf. Mobile Syst., Appl. Services*, New York, NY, USA, 2015, pp. 105–119.

[32] L. Lao and J. Cui, “Reducing multicast traffic load for cellular networks using ad hoc networks,” in *Proc. 2nd Int. Conf. Quality Service Heterogeneous Wired/Wireless Netw.*, Lake Vista, FL, USA, 2005, p. 10.

Authorized licensed use limited to: The Chinese University of Hong Kong CUHK(Shenzhen). Downloaded on May 18,2020 at 02:48:48 UTC from IEEE Xplore. Restrictions apply.

[33] B. Han, P. Hui, V. Kumar, M. Marathe, G. Pei, and A. Srinivasan,, "Cellular traffic offloading through opportunistic communications: A case study," in *Proc. 5th ACM Workshop Challenged Netw.*, New York, NY, USA, 2010, pp. 31–38.

[34] S. Kosta, A. Aucinas, P. Hui, R. Mortier, and X. Zhang, "ThinkAir: Dynamic resource allocation and parallel execution in the cloud for mobile code offloading," in *Proc. IEEE INFOCOM*, Orlando, FL, USA, Mar. 2012, pp. 945–953.

Fangyuan Chi (S'15) received the BASc degree in computer engineering from the University of British Columbia (UBC) in 2012 and the MASc degree from the University of British Columbia (UBC) in 2015. After graduated from UBC in 2012, she joined as a research and development engineer in the Department of Industrial Engineering, Tsinghua University, where she worked mainly on developing manufacturing execution system. She has published a paper in CloudCom2014, which had been selected for the Best Paper Award. She is a student member of the IEEE.

Xiaofei Wang (S'06-M'13) received the BS degree from the Department of Computer Science and Technology, Huazhong University of Science and Technology (HUST) in 2005, and the MS and PhD degrees from the School of Computer Science and Engineering, Seoul National University (SNU) in 2008 and 2013, respectively. He is currently a postdoctoral research fellow in the Department of Electrical and Computer Engineering, University of British Columbia (UBC). He received the Korean Government Scholarship for Excellent Foreign Students in the IT Field by NIPA from 2008 to 2011, and he also received the Global Outstanding Chinese PhD Student Award in 2012. His current research interests include social-aware multimedia service in cloud computing, cooperative backhaul caching, and traffic offloading in mobile content-centric networks. He is a member of the IEEE.

Wei Cai (S'12-M'XX) received the BEng degree from Xiamen University, Xiamen, China, in 2008 and the MSc degree from Seoul National University, Seoul, Korea, in 2011. He is currently working toward the PhD degree with the Department of Electrical and Computer Engineering, The University of British Columbia (UBC), Vancouver, Canada. He has completed visiting research at Academia Sinica, Taipei, Taiwan, The Hong Kong Polytechnic University, Hong Kong, and National Institute of Informatics, Tokyo, Japan. He has authored more than 10 first-author international journal/conference papers in gaming as a service, mobile cloud computing, online gaming, software engineering and interactive multimedia. He received many awards, such as UBC Doctoral Four-Year-Fellowship, the Brain Korea 21 Scholarship, and the Excellent Student Scholarship from Bank of China. He also co-received of Best Paper Awards from CloudCom2014, SmartComp2014, and CloudComp2013. He is a member of the IEEE.

Victor C.M. Leung (S'75-M'89-SM'97-F'03) received the BASc (Hons.) degree in electrical engineering from the University of British Columbia (UBC) in 1977, and was awarded the APEBC Gold Medal as the head of the graduating class in the Faculty of Applied Science. He attended graduate school at UBC on a Natural Sciences and Engineering Research Council Postgraduate Scholarship and completed the PhD degree in electrical engineering in 1981. From 1981 to 1987, he was a senior member of Technical Staff and satellite system specialist at MPR Teltech Ltd., Canada. In 1988, he was a lecturer in the Department of Electronics, the Chinese University of Hong Kong. He returned to UBC as a faculty member in 1989, and currently holds the positions of a professor and TELUS Mobility Research chair in Advanced Telecommunications Engineering in the Department of Electrical and Computer Engineering. He has coauthored more than 800 technical papers in international journals and conference proceedings, 30 book chapters, and co-edited 10 book titles. Several of his papers had been selected for Best Paper Awards. His research interests are in the areas wireless networks and mobile systems. He is a registered professional engineer in the Province of British Columbia, Canada. He was a distinguished lecturer of the IEEE Communications Society. He is a member of the editorial boards of the *IEEE Wireless Communications Letters*, *IEEE Journal on Selected Areas in Communications Series on Green Communications and Networking*, *Computer Communications*, and several other journals, and has previously served on the editorial boards of the *IEEE Journal on Selected Areas in Communications Wireless Communications Series*, *IEEE Transactions on Wireless Communications*, *IEEE Transactions on Vehicular Technology*, *IEEE Transactions on Computers*, and *Journal of Communications and Networks*. He has guest-edited many journal special issues, and provided leadership to the organizing committees and technical program committees of numerous conferences and workshops. He received the IEEE Vancouver Section Centennial Award and 2012 UBC Killam Research Prize. He is a fellow of the IEEE, the Royal Society of Canada, the Engineering Institute of Canada, and the Canadian Academy of Engineering.

▷ **For more information on this or any other computing topic, please visit our Digital Library at** www.computer.org/publications/dlib.

Authorized licensed use limited to: The Chinese University of Hong Kong CUHK(Shenzhen). Downloaded on May 18,2020 at 02:48:48 UTC from IEEE Xplore. Restrictions apply.