Delay-Sensitive Multiplayer Augmented Reality Game Planning in Mobile Edge Computing

Mike Jia
Research School of Computer Science
The Australian National University
Canberra, ACT, Australia
u5515287@anu.edu.au

Weifa Liang
Research School of Computer Science
The Australian National University
Canberra, ACT, Australia
wliang@cs.anu.edu.au

ABSTRACT

Mobile Edge Computing (MEC) is essential for enabling new innovative technologies that depend on low-latency computation environments such as Augmented Reality (AR). As AR applications continue to deliver better graphics with richer interactive features, AR devices will increasingly rely on nearby cloudlets to assist with the demanding computation requirements of AR applications. Supporting multiplayer interactions in an MEC environment brings many challenges. Processing user interactions can be computation-intensive especially when multiple users in close proximity to each other are acting simultaneously; the limited resources of a cloudlet could be overwhelmed if there are too many players involved. In this paper, we envision a scenario in the near future where players wearing AR heads-up display devices engage with other players over a large area with densely deployed cloudlets. We first propose a novel system model, and then formulate the Decentralized Multiplayer Coordination (DMC) Problem with the aim of minimizing the game frame duration among players, and devise an efficient algorithm for the problem. We finally evaluate the performance of the proposed algorithm through experimental simulations. Experimental results demonstrate that the proposed algorithm is promising.

KEYWORDS

Mobile edge computing; augmented reality; delay-sensitive task offloading; decentralized multiplayer game modelling; resource allocation and scheduling in MEC.

ACM Reference Format:
Mike Jia and Weifa Liang. 2018. Delay-Sensitive Multiplayer Augmented Reality Game Planning in Mobile Edge Computing. In *21st ACM International Conference on Modelling, Analysis and Simulation of Wireless and Mobile Systems (MSWIM '18), October 28-November 2, 2018, Montreal, QC, Canada.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3242102.3242129

1 INTRODUCTION

Mobile Edge Computing (MEC) has recently emerged as a key talking point in implementing the next generation of wireless cellular systems. MEC pushes cloud computing capabilities to the edge of the network by densely deploying cloudlets collocated with micro-base stations in urban areas [1]. By providing a reliable low latency computing environment for mobile users, MEC meets the requirements of new innovative technologies such as vehicle automation and augmented reality.

Augmented Reality (AR) is a technology that superimposes interactive digital elements on top of the real world view of a user device, and has attracted considerable investment from major technology companies. In 2017 at the F8 developer conference, Facebook CEO Mark Zuckerberg spoke at length about the potential of AR and described a future where artists could display digital artwork in public spaces and friends could share virtual signs and objects. AR could also disrupt the work environment, with AR headset displays like Microsoft Hololens and the Intel Vaunt smartglass being marketed as productivity tools that improve collaborations among colleagues. However, AR has been particularly successful in games, as demonstrated by the explosive popularity of the mobile AR game Pokemon Go. Pokemon Go was released in July 2016 and became the most active mobile game in the United States while generating more than 160 million US dollars through in-game purchases before the end of the month [16].

As AR applications continue to deliver better graphics and richer interactive features, the computation and network bandwidth demands of AR will increase. However, the computation resource of portable AR devices remain limited, and as a result AR devices will increasingly depend on cloudlets deployed throughout mobile networks to deliver cached contents and provide low latency computation environments. Since AR integrates digital elements into the real world, many AR applications and services will be specific to certain locations. In [5] Jia et. al., discuss how a local cloudlet could host *virtualized network functions* (VNFs) for processing AR service requests and serve multiple users in the same area. However, the authors did not consider the multiplayer interactions, and the additional challenges of a multiplayer system have yet to be addressed. Processing user interactions can be computation-intensive especially when multiple users in close proximity to each other are acting simultaneously; the limited resources of a cloudlet could be overwhelmed if there are too many players involved. To support a large number of players simultaneously, it is necessary that the workload of processing user interactions in an MEC network is evenly distributed among the cloudlets, to ensure that players receive feedback from their actions with short delay. However, coordinating a decentralized multiplayer system with large numbers of users is a challenge.

In this paper, we make the following contributions. We first envision a scenario in the near future where game players wearing

AR heads-up display devices engage with other players over a large area with densely deployed cloudlets, and introduce a novel system model for supporting a massive multiplayer game in AR. We then formulate the Decentralized Multiplayer Coordination (DMC) Problem with the aim of minimizing the game frame duration among players, and devise an efficient algorithm for the problem. We finally evaluate the performance of the proposed algorithm through experimental simulations. Experimental results demonstrate that the proposed algorithm is promising.

The remainder of the paper is organized as follows. Section 2 will survey state-of-the-arts of related topics in Augmented Reality, Massive Multiplayer Online Game Architectures, and Mobile Edge Computing. Section 3 will introduce the system model and formulate the problem, Section 4 will devise the algorithm for the problem, and Section 5 will provide the experimental results of the proposed algorithm. Finally, Section 6 will conclude the paper.

2 RELATED WORK

In this section we describe the unique requirements of gameplay in Augmented Reality, multiplayer game architectures, and previous studies on mobile edge networks.

Mobile Edge Computing (MEC) is a promising technology that brings high bandwidth, low latency computing environments close to mobile users. While cloud computing provides an abundance of computing resources for applications with intense computation demands, the internet delay between the remote cloud and users poses a serious issue for applications with sensitive delay tolerances [1, 11]. MEC pushes cloud computing capabilities to the edge of the mobile network, by deploying clusters of computers known as *cloudlets* within a Radio Area Network (RAN) close to mobile users [3, 5, 12, 13]. The close proximity of cloudlets to end users also enables the development of new services, such as Augmented Reality (AR).

Since AR combines virtual elements with real world environments, many AR games and objects will exist in the context of a specific environment, e.g., digital fish swimming in a real world fountain. These AR games and elements can thus be hosted on nearby cloudlets and accessed by users in the area who connect to the cloudlet [4, 10]. While some studies [2] have explored how MEC or similar technologies can be used to improve multiplayer games, these studies focused on mobile games that have different requirements to multiplayer AR games, and assumed that all player interactions take place on a central multiplayer game server in the cloud.

An AR device displays digital elements to a user by proceeding in *frames* [10]. At the start of a frame, an image is captured on the device's camera, and the user's precise position and orientation are aggregated from the raw data stream of sensors like accelerometers. The image frame may also be analyzed to identify surfaces, obstacles, landmarks that may effect the appearance or behavior of the digital elements. A view of the digital elements is then rendered according to the aggregated data and integrated with the captured image. The image is displayed to the user and the next frame begins. Real-time multiplayer games also proceed frame by frame, where actions performed by players are taken as input at the beginning of the game frame, and a sequence of events are generated as output at the end of each game frame. The duration of a game frame is of critical importance to the user experience, as it represents the length of time for a player to receive feedback from his or her action. Long and erratic game frames can irritate the player and potentially render the game unplayable.

There are two types of game architectures in multiplayer game systems: the traditional centralized game architecture; and the multi-server architecture [14]. In the former, a central coordinator also called a game server, receives player actions as input and generates a sequence of events as output. Its advantage is that the server has a complete picture of all player actions and can generate the authoritative sequence of events in the game. However generating events for player actions in richly interactive games is very computation-intensive, especially when there are a large number of players. Hosting the central coordinator on either a cloudlet in the mobile edge network or in the remote cloud has significant drawbacks. A cloudlet will have close proximity to the players and can provide low latency computation, but can only support a limited number of players due to its limited computing resource. On the other hand, the remote cloud has abundant computing resources but the latency between players and the remote cloud can lead to a poor game experience for players.

In the multi-server game architecture, multiple servers at different locations share the workload of processing player actions. A common approach to distributing this workload across multiple servers or cloudlets, is to partition players into different regions and assign a region coordinator from among the cloudlets to process player actions for each region [7, 9, 15]. The advantage is that the player's game experience benefits from the low latency to their region coordinators, while the workload of each region coordinator remains manageable. In [7, 15], both studies proposed that game developers first divide the game world into different regions (possibly with different features and challenges for each region) where the actions of players in each region are processed by a specific region coordinator. Knutsson et.al., assumed in [7] that while players can travel from one region to another via a hand-off from one server to another, players cannot interact with objects outside their regions. [15] improves upon this model by making use of a dynamic hierarchy to support user movement and interactions between different regions. A drawback of the fixed region assumption in [7, 15] is that players tend to flock to certain profitable areas of the game world, which can overwhelm the server or cloudlet that is coordinating the region. An alternative is to partition players into regions periodically, such that the workload of each region is roughly balanced.

Most existing studies assumed that each player is only able to interact with objects and players within a limited Area Of Interest (AOI) around the player. Fig. 1 is an illustrative example where the AOI of each player is illustrated by the dotted circle around him/her. Since players i and j are within each other's AOI, actions performed by i and j should be processed together as their actions may affect each other. However, since player i belongs to region r_2 and player j belongs to the region r_3, it is necessary for the region coordinators to exchange the player actions of i and j with each other to generate the correct sequence of events for players i and j. When deciding how to partition players into regions, it is necessary to consider the communication cost of exchanging player

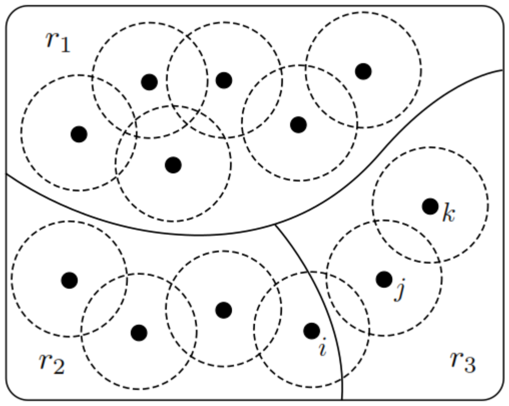


Figure 1: Partitioning players into groups with overlapping AOIs

data across partition boundaries. It can also be seen from Fig. 1 that the population of region r_1 is disproportionately larger than regions r_2 and r_3, and could potentially overwhelm the compute capacity of its region coordinator. Thus, there is clearly a trade-off between the computation workload of each region coordinator and the volume of communication between adjacent region coordinators when partitioning players into regions. This trade-off is addressed in [9], where its authors presented an algorithm that partitions players into a given number of regions, with the objective of minimizing the weighted sum of the computation cost incurred by the region coordinators and the communication cost between different region coordinators. While [9] dynamically partitions players into different regions, the study does not consider bandwidth constraints between region coordinators, and assumes that all servers have identical computation resources. Furthermore, their system model does not capture unique aspects of multiplayer AR, such as offloading routine but computation-intensive AR tasks from a player's device to a local cloudlet. In this work we address these issues and provide a detailed system model that accurately reflect the state of the art in MEC and AR.

3 SYSTEM MODEL

In this section, we first give an overview of the system model. We then formally define our objective, and then we finally define the Decentralized Multiplayer Coordination Problem.

3.1 Overview

We describe a system that can support the demanding bandwidth and computation needs of an AR multiplayer game. We assume that players wear AR heads-up display devices that overlay digital elements on the real world, while wearable sensors capture the player's physical actions and gestures. The game is real-time, where player actions are processed immediately, allowing players to fluidly interact with each other. Let there be K cloudlets $\{c_1, c_2...c_K\}$ which are densely deployed in the mobile edge network, a remote cloud denoted by c_0, N players $\{1, 2, ..., N\}$, and M regions $\{r_1, r_2, ..., r_M\}$, to which each player belongs to only one region. Each player wirelessly connects to a nearby cloudlet which

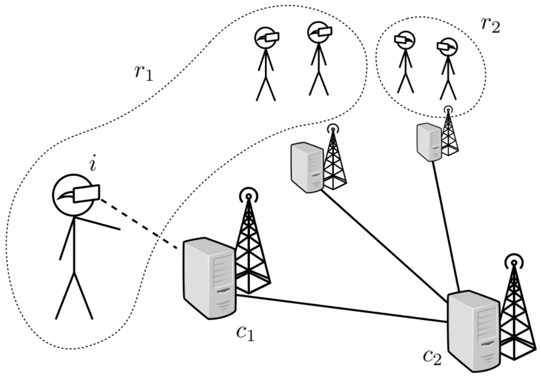


Figure 2: Overview of a game frame

manages the player's view of nearby game elements and communicates with the region coordinator that is responsible for managing the player's interaction with the other players in the region.

The multiplayer game system executes in a series of *game frames*, where each game frame begins by capturing the actions of each player, and concludes when all player actions have been processed by their region coordinators, which is illustrated in Fig. 2.

A game frame consists of three stages, which details as follows.

In stage one, the actions of player i are captured and uploaded to the player's connected cloudlet. The player's device and body sensors for detecting gestures must continuously collect large volumes of raw data in each game frame and process it to extract the player's aggregated *action data*, i.e. the player's location, movement, orientation, and intentions. Processing the raw data into action data is computation-intensive, and offloading the computation from the player device to the connected cloudlet can reduce the duration of each game frame, improving user experience. Since raw data typically has a larger volume than the action data, there is a non-trivial tradeoff between the time spent wirelessly uploading raw data directly to the cloudlet, and the time spent processing the raw data on the local device.

In stage two, the action data of players in each region needs to be transmitted from the player's connected cloudlet to the region coordinator. We assume that a cloudlet may only serve as the region coordinator for at most one region, and the remote cloud may be appointed as the region coordinator for some regions if there is insufficient computing resources among cloudlets. Let P_k be the set of players in the region r_k, $1 \leq k \leq M$. As seen in Fig. 2, players in region r_1 are close enough to players from region r_2 to interact with them. As a result, the region coordinator of r_1 must take into account the action data from these players outside r_1, as their actions could affect the players in r_1. To this end, we assume that the region coordinator of a given region r_k maintains a complete set of players P_k^+ that are within its scope from which it must receive action data, where $P_k \subseteq P_k^+$(recall that P_k denotes the set of players in region r_k). Stage two of the game frame for players in region r_k concludes once the connected cloudlets of players in P_k^+ have transmitted the action data of the players to the region coordinator of r_k. We assume that cloudlet-to-cloudlet communication is implemented through densely connected optic fiber cables, and the link between two given cloudlets c_i and c_j has a limited bandwidth $B(c_i, c_j)$ and a short delay $l(c_i, c_j)$ when transmitting player action data. As a result, the cloudlet locations of the region coordinators is important to ensuring a short game frame duration for players and avoiding congestion between cloudlets in the network.

In stage three, the region coordinator of r_1 can process the action data of the players, and transmit the outcome back to the players in the region (players in region r_2 receive updates from the region coordinator of r_2, not r_1). The duration of each stage will vary from player to player and region to region, but the game frame concludes only when all region coordinators have finished processing the action data and all players have received updates from their region coordinator. We assume that the game frame duration is universal for all players to avoid the situation where some players are privileged with a shorter game frame duration, giving them an advantage over other players with a longer game frame duration.

The duration of each game frame is of critical importance to the user experience; noticeable latency between a player action and the resulting event occurring in the game is distracting and can produce feelings of nausea. As a result, special care must be taken in planning player-to-cloudlet connections, the offloading of critical data processing tasks from player devices to their connected cloudlets, and the positions of region coordinators, to fully optimize the duration of the game frame and to provide a positive game experience for all players.

3.2 Objective Definition

As our objective is to minimize the duration of each game frame, we now detail the duration of each stage in a game frame.

In stage one, player i performs a gesture, which is captured by the player's body sensors as *raw data* and is converted into action data. Let o_i be the average volume of raw data collected from player i per game frame, while the action data of player i has an average volume o'_i. Suppose the wireless transmission rate between player i and its connected cloudlet c_j is w_{ij}. Let vector variable $X = \{x_1, x_2, ..., x_N\}$ indicate the wireless connections between users and cloudlets, where x_i is the index of the cloudlet to which player i is connected, i.e., if player i is connected to cloudlet c_j, then $x_i = j$, for all i and j with $1 \le i \le N$ and $1 \le j \le K$. Let $d_L(i,j)$ and $d_O(i,j)$ denote the delay of processing the player's raw data at the local device and the delay of offloading the raw data and processing it remotely at cloudlet c_j, respectively. Let vector variable $Y = \{y_1, y_2, ..., y_N\}$ indicate the decision to process the raw data of each player on the player's local device or on a cloudlet, where $y_i = 0$ indicates the raw data of player i will be processed locally and $y_i = 1$ indicates the raw data will be processed on the player's connected cloudlet.

We assume that the processing delay can be modelled with an *M/M/*1 queue. Let μ_i^u and μ_j^c be the service rates for a unit of computation on the local device of player i and its connected cloudlet c_j, respectively, and let λ_P denote the units of computation required to process each unit of raw data. When the raw data is processed into action data on the local device, then the delay $d_L(i,j)$ between player i and its connected cloudlet c_j is defined as the sum of the transmission time for the processed action data, the queuing time on the player's device and the computation time, i.e.,

$$d_L(i,j) = \left(\frac{o'_i}{w_{ij}} + \frac{1}{\mu_i^u - o_i \cdot \lambda_P} + \frac{o_i \cdot \lambda_P}{\mu_i^u}\right), \tag{1}$$

where w_{ij} is the data rate between the player and the connected cloudlet. By offloading the task of raw data processing to the cloudlet, this increases the workload of the cloudlet.

We then consider the case when the player's raw data is offloaded to a cloudlet. Let $\lambda_T(c_j)$ denote the workload of cloudlet c_j. As cloudlet c_j may also be serving as a region coordinator, we formally define the total workload $\lambda_T(c_j)$ of c_j later, in Eq. (9). Recall that $d_O(i,j)$ denotes the delay for offloading the raw data of player i and processing it on the connected cloudlet c_j. $d_O(i,j)$ can be calculated as the sum of the transmission time of the raw data, the queuing time on the cloudlet and the computation time, that is,

$$d_O(i,j) = \left(\frac{o_i}{w_{ij}} + \frac{1}{\mu_j^c - \lambda_T(c_j)} + \frac{o_i \cdot \lambda_P}{\mu_j^c}\right). \tag{2}$$

We can then calculate the stage one time $t_1(i)$ of the game frame for player i, that is the time taken for the action data of player i to be available on the cloudlet:

$$t_1(i) = (1 - y_i)\, d_L(i, x_i) + y_i \cdot d_O(i, x_i). \tag{3}$$

In stage two, the action data needs to be transmitted to the player's region coordinator. Let vector variable $Z = \{z_1, z_2, ..., z_M\}$ denote the location of the region coordinator for each region, where z_k is the unique index of the remote cloud or cloudlet location for the coordinator of region r_k. Recall P_k^+ denotes the set of all players within the scope of the region coordinator of r_k, and $l(c_i, c_j)$ denotes the latency between cloudlets c_i and c_j. We can then calculate the stage two time $t_2(r_k)$ for players belonging to the region r_k, as the maximum of the time taken to reach the end of stage one for each individual player in P_k^+ (given in Eq. (3)), plus the data transmission delay from the player's connected cloudlet to the region coordinator:

$$t_2(r_k) = \max_{h \in P_k^+} \{t_1(h) + l(x_h, z_k)\}. \tag{4}$$

Note that $t_2(r_k)$ covers the duration of the stage one time $t_1(i)$ for all players i within the scope of region r_k, and measures the total time taken from the beginning of the game frame to the end of stage two for region r_k.

Recall that there is a bandwidth limit $B(c_i, c_j)$ between cloudlets c_i and c_j. As data transfer occurs between cloudlets directly connected to players and cloudlets serving as region coordinators, we calculate the total amount of data $b(c_i, c_j)$ transferred from a connected cloudlet c_i to the cloudlet c_j which is the region coordinator of r_k:

$$b(c_i, c_j) = \sum_{h \in U_j \cap P_k^+} o'_h, \tag{5}$$

where U_j is the set of players connected to cloudlet c_j, i.e. $U_j = \{\, i \mid x_i = j,\ x_i \in X \,\}$. We then have the following bandwidth constraint:

$$b(c_i, c_j) \le B(c_i, c_j),\ \forall 1 \le i, j \le K. \tag{6}$$

We assume there is no broadband limit for transmitting player data from a connected cloudlet to a region coordinator located in the remote cloud, since cloudlets make use of the backhaul network to connect to the remote cloud via the Internet.

In stage three, each region coordinator must process the player action data. The compute requirement of each region coordinator to process player actions depends on the total volume of the action data to be processed.

Let $\tau_k(c_j)$ denote the total computing delay of processing the player action data of the region coordinator of r_k at location c_j. If the region coordinator of r_k is assigned to the remote cloud c_0, then computing resources are effectively unlimited and we can assume a fixed processing delay in the cloud d_C for processing the player action data for players in r_k, that is,

$$\tau_k(c_0) = d_C.$$

However, if the region coordinator has been assigned to a cloudlet, then we must take into account the limited cloudlet resources when calculating the computing delay.

Let $\Psi(\cdot)$ be a non-decreasing function of the amount of computation units needed to process action data with its variable parameter being a set of players, and recall that we denote by P_k^+ the set of players within the scope of region r_k, μ_j^c is the service rate of cloudlet c_j, and $\lambda_T(c_j)$ is the total workload on cloudlet c_j. The total computing delay $\tau_k(c_j)$ to process the player action data of the region coordinator of r_k on cloudlet c_j consists of the queuing time on cloudlet c_j and the computation time, which is defined as follows.

$$\tau_k(c_j) = \frac{1}{\mu_j^c - \lambda_T(c_j)} + \frac{1}{\mu_j^c} \cdot \Psi\left(P_k^+\right), \tag{7}$$

where $1 \leq j \leq K$. We now formally define the total workload of the cloudlet $\lambda_T(c_j)$. Recall that players connected to cloudlet c_j may offload their raw data to c_j. Let O_j denote the total volume of raw data to be processed by cloudlet c_j:

$$O_j = \sum_{i \in U_j} y_i \cdot o_i', \tag{8}$$

recall that U_j is the set of players connected to cloudlet c_j, i.e. $U_j = \{\, i \,|\, x_i = j,\ x_i \in X \,\}$. The workload $\lambda_T(c_j)$ of cloudlet c_j can thus be defined as the sum of the computation required to process the volume of raw data O_j in stage one and (if the cloudlet c_j is the region coordinator of a given region r_k) the computation required to process the action data of players in region r_k in stage three:

$$\lambda_T(c_j) = O_j \cdot \lambda_P + \Psi\left(P_k^+\right). \tag{9}$$

We can then calculate the stage three time $t_3(r_k, z_k)$ for all players belonging to the region r_k as the sum of the computing delay at location z_k and the stage two time for players in region r_k given in Eq. (4):

$$t_3(r_k, z_k) = \tau_k(z_k) + t_2(r_k) \tag{10}$$

The stage three time for each region represents the time taken for a player in the region to receive feedback from his/her actions. This can give an advantage to players from regions that reach the end of stage three earlier than other regions. In the interest of fairness, we assume that the duration of the game frame $T(X, Y, Z)$ is the maximum time taken among all regions to reach the end of stage three. Thus we define the objective function $T(X, Y, Z)$:

$$T(X, Y, Z) = \max_{1 \leq k \leq M} t_3(r_k, z_k). \tag{11}$$

3.3 Problem Definition

Given an MEC network consisting of K cloudlets and N players with each player belonging to a specified region, *the Decentralized Multiplayer Coordination (DMC) Problem* is to minimize the objective in Eq. (11) (i.e., $\min_{X,Y,Z} T(X, Y, Z)$), by deciding the three vector variables X, Y, and Z while meeting the network bandwidth constraint in Eq. (6), where the vector variable $X = \{x_1, x_2, \ldots, x_N\}$ decides which cloudlet to connect to each player i, the vector variable $Y = \{y_1, y_2, \ldots, y_N\}$ decides whether the action data of player i is extracted on its local device or on its connected cloudlet, and the vector variable $Z = \{z_1, z_2, \ldots, z_M\}$ decides a cloudlet or the remote cloud to be appointed as the region coordinator of region r_k.

4 ALGORITHM

In this section, we present an iterative algorithm for the DMC problem. We first assign each user to the cloudlet with which it has the strongest wireless connection. Each player has a number of nearby cloudlets it can connect to, and recall that w_{ij} is the wireless transmission rate between player i and a given cloudlet c_j. We thus connect player i to cloudlet c_j with the largest data rate, i.e., we set $x_i = \arg\max_{1 \leq j \leq K} w_{ij}$ for each player i. We then set the initial location of each region coordinator to be the remote cloud due to the abundance of computing resources, i.e., $z_k = 0$, $1 \leq k \leq M$. Once we have initial values for vector variables X and Z, we can assign values to the vector variable Y according to the following procedure.

Since most players connected to the same cloudlet will have their action data sent to the same region coordinator, the player with the longest stage one time may be the bottleneck for the entire region. Recall that vector variable Y decides the offloading of raw data for each player to their connected cloudlet, and U_j is the set of players connected to cloudlet c_j according to vector variable X. To ensure that region coordinators receive player action data from the players within their scopes as soon as possible, we determine vector variable Y to optimize the following objective:

$$\min_{\{y_i | i \in U_j\}} \max_{i \in U_j} t_1(i), \tag{12}$$

where $y_i = \{0, 1\}$, $\forall i \in U_j$. However, not all players may benefit from offloading their raw data to the cloudlet. If the volume of raw data o_i for player i is particularly large and the player has sufficient computing resources on its local device, offloading the player's raw data may result in a longer stage one time than if the player's raw data was processed locally. Recall that $d_O(i)$ and $d_L(i)$ denote the duration of stage one if the raw data of player i is offloaded, and if the raw data is processed locally, respectively. For each player connected to cloudlet c_j, we first compare $d_L(i)$ with $d_O(i)$, where player i is the only player offloading to the cloudlet. If the local processing time $d_L(i, j)$ is lower than $d_O(i, j)$, even when player i

is the sole player offloading to its connected cloudlet, then player i has a preference for local processing, and we set $y_i = 0$.

Procedure 1 decideOffloading(X, Z)

Decide offloading of player raw data given player-cloudlet connections and region coordinator locations

```
Require: $X$, $Z$
Ensure: $Y$.
 1: /* For each cloudlet with connected players */
 2: for $j \leftarrow 1$ to $K$ do
 3:     /* Initially set players to process raw data locally */
 4:     for $i \in U_j$ do
 5:         $y_i \leftarrow 0$
 6:     $U \leftarrow U_j$
 7:     /* Remove players that prefer local processing from $U$ */
 8:     for $i \in U_j$ do
 9:         $y_i \leftarrow 1$
10:         if $d_O(i, j) > d_L(i, j)$ then
11:             $U \leftarrow U - \{i\}$;
12:         $y_i \leftarrow 0$;
13:     Add remaining players $U$ to priority queue $Q$ according to max $d_L(i, j)$;
14:     while $Q \neq \emptyset$ do
15:         $i' \leftarrow Q$.removeHead();
16:         $t_1 \leftarrow \max_{h \in U_j} t_1(h)$
17:         $y_{i'} \leftarrow 1$
18:         if $t_1 < \max_{h \in U_j} t_1(h)$ then
19:             $y_{i'} \leftarrow 0$
20:             exitLoop();
21:     Return $Y$;
```

To decide the offloading decision for the remaining players, we form a priority queue from the remaining players according to their local processing time $d_L(i, j)$, such that the head of the queue is the player with $\max_{i \in U_j} d_L(i, j)$. We take player i at the head of the queue and set $y_i = 1$ to offload its raw data to connected cloudlet c_j. Since player i is the first player to offload raw data to the cloudlet, the offloading time $d_O(i, j)$ will be smaller than $d_L(i, j)$, decreasing the maximum stage one time for players in U_j according to our objective in (Eq. (12)). However, this may not be the case when offloading the raw data of subsequent players in the queue, since the queuing time at the cloudlet will increase as more players offload their raw data. Thus for each subsequent player i' we remove from the queue, we first compare the objective $\max_{h \in U_j} t_1(h)$ when player i' processes its raw data locally, and when the raw data of player i' is offloaded (i.e., we set $y_{i'} = 1$). If $\max_{h \in U_j} t_1(h)$ is greater when the current player i' offloads raw data compared to when the raw data is locally processed, this indicates that further offloading player raw data will not improve our objective, and we let player i' and the remaining players in the queue process their raw data locally. Otherwise, we let player i' offload its raw data, and remove the next player from the queue. The details of this procedure are given in Procedure 1.

Once we have initial values for variables X, Y, and Z, we iteratively refine the solution by focusing on regions and players that are the bottlenecks in the system. We start by identifying the region r_k with the longest stage three time $t_3(r_k, z_k)$ (defined in Eq. (10)), as it dominates the duration of the game frame. Since the location of each region coordinator has a great impact on the stage three time

Algorithm 1 *IterativeAlgorithm*

```
Require: $o_i$, $o'_i$, $\mu^u_i$, $i \in \{1...N\}$, $\mu^c_j$, $j \in \{1...K\}$, $P^+_k$, $k \in \{1...M\}$, MAX.
Ensure: $X$, $Y$, $Z$
 1: /* Connect each player $i$ to cloudlet with strongest connection */
 2: $X \leftarrow \{x_i \leftarrow \arg\max_{1 \leq j \leq K} w_{ij} \,|1 \leq i \leq N\}$;
 3: /* Set remote cloud to be region coordinator for all regions */
 4: $Z \leftarrow \{z_k \leftarrow 0 \,|1 \leq k \leq M\}$;
 5: /* Obtain $Y$ by invoking Procedure 1 */
 6: $Y \leftarrow$ decideOffloading($X$, $Z$);
 7: $t_{max} \leftarrow \infty$;
 8: iter $\leftarrow 0$;
 9: while $t_{max} > T(X, Y, Z)$ (defined in Eq. (11)) and iter $<$ MAX do
10:     iter $\leftarrow$ iter $+ 1$;
11:     $t_{max} \leftarrow T(X, Y, Z)$;
12:     $r_k \leftarrow \arg\max_{1 \leq k \leq M} t_3(r_k)$;
13:     /* Find alternative location for region coordinator of $r_k$ */
14:     $z^\star_k \leftarrow \arg\min_{0 \leq j \leq K} t_3(r_k, j)$ and bandwidth constraint Eq. (6) holds;
15:     $Z \leftarrow Z - \{z_k\} + \{z^\star_k\}$
16:     if $z^\star_k = z_k$ then
17:         /* Identify the bottleneck player in region $r_k$ */
18:         Let $i \leftarrow \arg\max_{h \in P^+_k} \{t_1(h) + l(x_h, z_k)\}$;
19:         /* Find alternative cloudlet for player $i$ to connect to */
20:         for $j \leftarrow 1$ to $K$ and $w_{ij} > 0$ do
21:             $x'_i \leftarrow j$;
22:             $X' \leftarrow X - \{x_i\} + \{x'_i\}$;
23:             /* Obtain $Y'$ by invoking Procedure 1 */
24:             $Y' \leftarrow$ decideOffloading($X'$, $Z$);
25:             if $T(X', Y', Z) < t_{max}$ and bandwidth constraint Eq. (6) holds then
26:                 $t_{max} \leftarrow T(X', Y', Z)$;
27:                 $X^\star \leftarrow X'$ and $Y^\star \leftarrow Y'$;
28:         $X \leftarrow X^\star$ and $Y \leftarrow Y^\star$;
29: Return $X$, $Y$, $Z$;
```

of the region, we begin by finding an alternative location $z^\star_k$ for the region coordinator of r_k, where $z^\star_k$ is the location that minimizes the stage three time of region r_k:

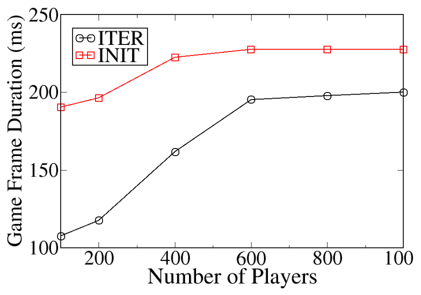
$$z^\star_k = \arg\min_{0 \leq j \leq K} t_3(r_k, j),$$

while observing the bandwidth constraint in Eq. (6). If $z^\star_k = z_k$, this means that the stage three time of region r_k cannot be improved by changing the location of the region coordinator. In this case, it is still possible to reduce $t_3(r_k, z_k)$ by reducing the stage two time of region r_k.

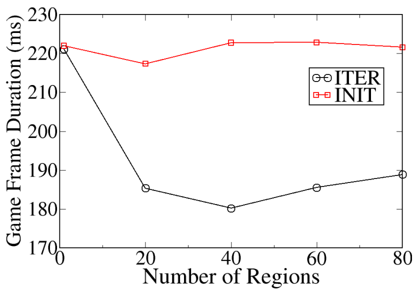
In stage two, the region coordinator of r_k must wait to receive the action data of all players within its scope, and a single player could bottleneck the entire region. We thus identify the bottleneck player i for region r_k as the player whose action data arrives the latest at the region coordinator:

$$i = \arg\max_{h \in P^+_k} \{t_1(h) + l(x_h, z_k)\}, \tag{13}$$

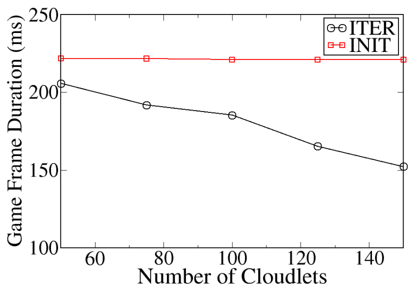
where we recall that P^+_k is the set of all players within the scope of r_k, and $l(x_h, z_k)$ is the latency between the connected cloudlet x_h and region coordinator location z_k. By finding an alternative cloudlet for the bottleneck player i to connect to, we can reduce the stage two time of region r_k thus reducing the duration of the



(a) Number of Players

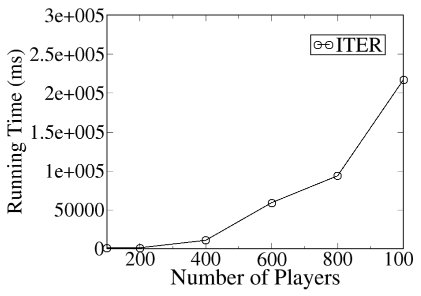


(b) Number of Regions

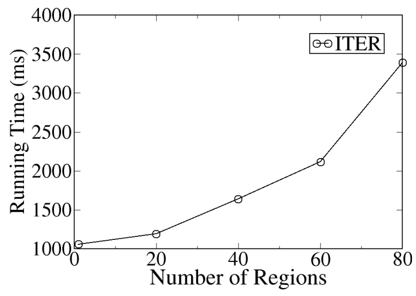


(c) Number of Cloudlets

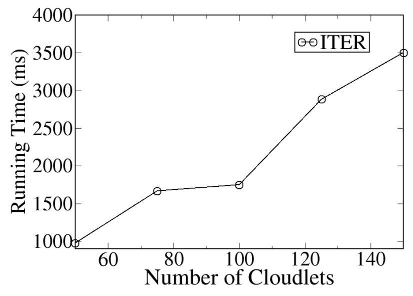
Figure 3: The performance of the proposed algorithm and the benchmark.



(a) Number of Players



(b) Number of Regions



(c) Number of Cloudlets

Figure 4: The running time of the proposed algorithm.

game frame. For each potential alternative cloudlet c_j, we set $x_i = j$ and we re-optimize Y using Procedure 1. We then connect player i to the cloudlet that yields the greatest reduction in game frame duration and does not violate the bandwidth constraint in Eq. (6). If no changes can be made to improve the duration of the game frame, then the final assignment to variables X, Y, and Z returns a solution to the problem. Algorithm 1 gives the details.

5 PERFORMANCE EVALUATION

In this section we evaluate the performance of Algorithm 1. We start with the environment setting and then conduct the performance evaluation.

To generate a simulation environment, we randomly position 100 cloudlets and 500 players on a 1,000 meter square grid. If a given player i is very close to a cloudlet c_j, the wireless data rate w_{ij} between them will be 1,000 *Mbps*, and w_{ij} is inversely proportional to the physical distance between player i and cloudlet c_j. Players are partitioned into 20 regions according to the algorithm adapted from [9]. We determine the set of players P_k^+ within the scope of the region coordinator of r_k, by including all players outside the region r_k that fall within an Area of Interest (AOI) radius of 50 meter of players within r_k. We assume that the latency between any two cloudlets is on average 2 to 5 milliseconds, while bandwidth limit between cloudlets is drawn randomly from the range 1,000 *Mbps* to 10,000 *Mbps* [6, 8]. Finally we assume the total computation to be processed by a region coordinator is quadratic with with the number of players whose actions are being processed. The running time of the proposed algorithm is obtained based on a machine with a 4 GHz Intel i7 Quad-core CPU and 32 GiB RAM. The unit of running time is shown in milliseconds. Unless otherwise specified these parameters will be adopted in the default setting.

We compare the performance of the proposed algorithm `iter` with a benchmark `init` where we take the initial solution in our proposed algorithm before it is iteratively improved, i.e., we connect player i to cloudlet c_j with the maximum data rate, i.e., we set $x_i = \arg\max_{1\le j\le K} w_{ij}$ for each player i. We set the initial location of each region coordinator to be the remote cloud due to the abundance of computing resources, i.e., $z_k = 0$, $1 \le k \le M$, and we solve Y using Procedure 1.

Fig. 3 (a) shows the relationship between the game frame duration and the number of players in the system. When the number of players increases from 100 to 200, the game frame duration delivered by the proposed algorithm `iter` slightly increases, as the cloudlets have more than enough resources to support the low number of players. However, when the number of players increases from 200 to 600, the game frame duration dramatically increases. Since the number of regions has not changed, the workload of each region coordinator increases sharply, and more region coordinators are assigned to the remote cloud, where computing resources are abundant. The game frame duration eventually plateaus when there are 1,000 players as most region coordinators have been assigned to the remote cloud. The game frame duration delivered by the benchmark `init` similarly increases with an increase in the number of players, as additional players are unable to offload their raw data to their connected cloudlet for processing. However, the game frame duration delivered by `init` quickly plateaus with a higher game frame duration compared to `iter` when the number of players is only 400. While both algorithms eventually assign all region coordinators to the remote cloud, `iter` delivers a smaller game frame duration as a result of optimizing the stage two time for some regions.

Fig. 3 (b) plots the game frame duration delivered by `iter` and `init` by changing the number of regions. At first when there is only one region, no cloudlet is able to handle the huge amount of computation required to process every player's action data and so the region coordinator is assigned to the remote cloud. As a result both `iter` and `init` deliver the same game frame duration. As the number of regions increases, the workload of each region coordinator dramatically decreases allowing `iter` to assign region coordinators among the cloudlets. The game frame duration delivered by `iter` reaches a minimum when the number of regions is 40. When the number of regions increases from 40 to 80, the game frame duration increases slightly due to the increased network traffic from connected cloudlets to the additional regions. As bandwidth between cloudlets is limited, some region coordinators are forced to be assigned to the remote cloud, increasing the game frame duration.

Fig. 3 (c) displays the game frame duration delivered by the proposed algorithm and the benchmark by increasing the number of cloudlets. The game frame duration delivered by `iter` decreases linearly as the number of cloudlets increase. As more cloudlets are added to the system, players increasingly offload raw data to the cloudlet as fewer players need to share a single connected cloudlet. Furthermore as more cloudlets are added, the average distance between a cloudlet and a player decreases, resulting in stronger connections and increased wireless data rate between players and connected cloudlets. The game frame duration delivered by the `init` also decreases linearly with the number of cloudlets, but at a lower gradient compared to that of the proposed algorithm.

Finally, Fig. 4 (a), (b) and (c) shows the running time of the proposed algorithm against the number of players, the number of regions and the number of cloudlets, respectively. As can be seen, the running time of the proposed algorithm is roughly linear with the number of cloudlets, while the running time of the proposed algorithm increases dramatically with the number of players and the number of regions.

6 CONCLUSION

In this paper, we envisioned a scenario in the near future where players wearing AR heads-up display devices engage with other players over a large area in an MEC network with densely deployed cloudlets, and introduced a novel system model for supporting a massive multiplayer game in AR. We then formulated the Decentralized Multiplayer Coordination (DMC) Problem with the aim of minimizing the game frame duration of all players, and also devised an efficient algorithm for the problem. We finally evaluated the performance of the proposed algorithm through experimental simulations. Experimental results demonstrated that the proposed algorithm is promising.

As AR technology and applications continue to improve and gain consumers, the proposed system model and algorithm provided in this paper can serve as a baseline for future studies of AR multiplayer systems. Several research issues still remain open, such as gracefully handling stochastic traffic conditions in the MEC network, which we leave to future works.

REFERENCES

[1] Sharad Agarwal, Matthai Philipose, and Paramvir Bahl. Vision: The case for cellular small cells for cloudlets. In *Proceedings of the fifth international workshop on Mobile cloud computing & services*, pages 1–5. ACM, 2014.

[2] Wei Cai, Victor CM Leung, and Long Hu. A cloudlet-assisted multiplayer cloud gaming system. *Mobile Networks and Applications*, 19(2):144–152, 2014.

[3] Mike Jia, Jiannong Cao, and Weifa Liang. Optimal cloudlet placement and user to cloudlet allocation in wireless metropolitan area networks. *IEEE Transactions on Cloud Computing*, vol.5, no.5 pp.725–737, 2017.

[4] Mike Jia, Weifa Liang, and Zichuan Xu. Qos-aware task offloading in distributed cloudlets with virtual network function services. In *Proceedings of the 20th ACM International Conference on Modelling, Analysis and Simulation of Wireless and Mobile Systems*, pages 109–116. ACM, 2017.

[5] Mike Jia, Weifa Liang, Zichuan Xu, Meitian Huang, and Yu Ma. Qos-aware cloudlet load balancing in wireless metropolitan area networks. To appear in *IEEE Transactions on Cloud Computing*, 2018.

[6] Simon Knight, Hung X Nguyen, Nick Falkner, Rhys Bowden, and Matthew Roughan. The internet topology zoo. *IEEE Journal on Selected Areas in Communications*, 29(9):1765–1775, 2011.

[7] Bjorn Knutsson, Honghui Lu, Wei Xu, and Bryan Hopkins. Peer-to-peer support for massively multiplayer games. In *INFOCOM 2004. Twenty-third AnnualJoint Conference of the IEEE Computer and Communications Societies*, volume 1. IEEE, 2004.

[8] Diego Kreutz, Fernando MV Ramos, Paulo Esteves Verissimo, Christian Esteve Rothenberg, Siamak Azodolmolky, and Steve Uhlig. Software-defined networking: A comprehensive survey. *Proceedings of the IEEE*, 103(1):14–76, 2015.

[9] John C. S. Lui and MF Chan. An efficient partitioning algorithm for distributed virtual environment systems. *IEEE Transactions on parallel and distributed systems*, 13(3):193–211, 2002.

[10] BG Rodriguez-Santana, Amilcar Meneses Viveros, Blanca Esther Carvajal-Gámez, and Diana Carolina Trejo-Osorio. Mobile computation offloading architecture for mobile augmented reality, case study: Visualization of cetacean skeleton. *Int. J. Adv. Comput. Sci. Appl*, 1(7):665–671, 2016.

[11] Mahadev Satyanarayanan, Paramvir Bahl, Ramón Caceres, and Nigel Davies. The case for vm-based cloudlets in mobile computing. *IEEE pervasive Computing*, 8(4), 2009.

[12] Zichuan Xu, Weifa Liang, Wenzheng Xu, Mike Jia, and Song Guo. Capacitated cloudlet placements in wireless metropolitan area networks. In *Local Computer Networks (LCN), 2015 IEEE 40th Conference on*, pages 570–578. IEEE, 2015.

[13] Zichuan Xu, Weifa Liang, Wenzheng Xu, Mike Jia, and Song Guo. Efficient algorithms for capacitated cloudlet placements. *IEEE Transactions on Parallel and Distributed Systems*, vol.27, no.10, pp.2866-2880, 2016.

[14] Amir Yahyavi and Bettina Kemme. Peer-to-peer architectures for massively multiplayer online games: A survey. *ACM Computing Surveys (CSUR)*, 46(1):9, 2013.

[15] Anthony P Yu and Son T Vuong. Mopar: a mobile peer-to-peer overlay architecture for interest management of massively multiplayer online games. In *Proceedings of the international workshop on Network and operating systems support for digital audio and video*, pages 99–104. Acm, 2005.

[16] SensorTower. Sensor Tower pokemon go worldwide revenue, 2016.