# Environment Perception for Cognitive Cloud Gaming

Wei Cai[1], Conghui Zhou[2], Victor C.M. Leung[1], and Min Chen[3]

[1] The University of British Columbia, Canada
{weicai,vleung}@ece.ubc.ca,
[2] We Software Limited, Hong Kong
neio.zhou@gmail.com,
[3] Huazhong University of Science and Technology, China
minchen2012@hust.edu.cn

**Abstract.** Mobile cloud games utilizes the rich resources within the cloud to enhance the functionality of mobile devices, therefore, to overcome the intrinsic constraints of mobile devices. To provide a satisfying quality of experience for players in dynamic network context, we need a cognitive gaming platform which is cognitive of resources and characteristics of the cloud, the access network, and the end-user devices, and enables dynamic utilization of these resources. In this work, we develop an environment perception solution with a novel capability to learn about the game player's environment (i.e., the combination of terminal and access network) to facilitate this cognitive gaming platform.

## 1 Introduction

Mobile game already has a large share of application market. However, the hardware constrains of mobile devices, such as the limited storage, insufficient computational capacity, and battery drain problems, restrict the design of mobile games. To address these design issues, mobile cloud computing [1] provides a potential solution. Well recognized as the next generation computing infrastructure, the cloud is considered to provide unlimited storage and computational resources, which supports various types of online services for cloud users. With the approach of offloading [2], the cloud-based game for mobile devices utilizes the rich resources within the cloud to enhance the functionality of mobile devices and prolong the battery lifetime through better energy efficiency, therefore, to overcome the intrinsic constraints of mobile devices.

Industry is now leading the way to commercialize the Gaming as a Service (GaaS). Companies, such as OnLive[1], Gaikai[2] and G-Cluster[3] are the most famous commercial providers of gaming services on-demand. In their remonte rendering GaaS provision model, the video games are executed in their private cloud servers and the generated game video frames are transmitted to the mobile client through Internet after the encoding process. In reverse, the players' inputs are delivered to the cloud and accepted by the game content server directly [3]. In this context, the cloud is intrinsically an interactive video generator and streaming server, while the mobile devices serve as the

[1] http://www.onlive.com
[2] http://www.gaikai.com
[3] http://www.g-cluster.com

event controllers and video receivers. Since the games are rendered in cloud servers, a less capable computer or mobile device may be used to support sophisticated games. This approach results in longer battery life for the device and longer gaming times for the user at the expense of higher consumption of communication resources. However, the remote rendering GaaS suffers from the bottleneck of Internet bandwidth, which constrains the bit rate of gaming videos, while the jitter and delay affect the quality of experience (QoE) for the players [4]. Therefore, the QoE-oriented adaptive gaming video rendering and transmission become the most promising research topic in this area. [5] proposes a set of application layer optimization techniques to ensure acceptable gaming response time and video quality in the remote server based approach. The techniques include downlink gaming video rate adaptation, uplink delay optimization, and client play-out delay adaptation.

With the recent development of hardware performance, most gaming terminals, including mobile devices, are capable to perform complicated graphical rendering for game scenes. Therefore, a local rendering GaaS model become practical. As a matter of fact, the promising HTML5 browser games fall into this category. Accordingly, a more flexible solution for the provision of GaaS is suggested by [6]. In this work, cloud-based games are modeled as inter-dependant components, which work collaboratively to provide gaming services for the players. During the gaming session, the cloud intelligently transmits selected game components to the users' terminal and performs a cognitive resource allocation between the cloud and user end terminals, on the purpose of system optimization while guaranteeing QoE for players. It can be considered as an iterative 3 phases procedure: 1) to predict the resource consumption of each game component and the communication cost between these components; 2) to perform efficient and accurate real-time measurements, evaluation, and prediction on cloud performance, access network performance and end-user device status; 3) to design joint adaptive strategy to optimize the system performance.

In summary, these two approaches both envision a flexible and feasible solution that is cognitive of resources and characteristics of the cloud, the access network, and the end user devices. As a cognitive system, they are required to be capable of i) **environment perception** to collect players' environmental data and to monitor the real-time system status; ii) **system analysis** to evaluate the system performance and to make decisions for the service provision; iii) **QoE-oriented adaption** to take actions for overall system optimization. In this work, we focus on **environment perception** and develop a data collecting solution with a novel capability to learn about the game player's environment (i.e., the combination of terminal and access network). To the best of our knowledge, this is the first work on the design and implementation of environment perception solution for cognitive cloud gaming, and it is also the very first work on the application of mobile agent [7] on this topic. The remaining sections of the paper is organized as follows. We review related work in Section 2 and describe our design of the environment perception in Section 3. Afterwards, in Section 4, we describe our implementation details. Concluding remarks are presented in Section 5.

## 2 Related Work

### 2.1 QoE for Cloud Gaming

The goal of environment perception is to collect the data that are have most impact on players' QoE. To provide GaaS, the relationships between cloud gaming QoE and QoS are different for distinct implementation architectures. For remote rendering GaaS, various subjective user studies have been conducted to demonstrate the relationship between cloud gaming QoE and QoS, including game genres, video encoding factors, CPU load, memory usage, and link bandwidth utilization [8], response latency, and the game's real-time strictness [9], number of users [10], network characteristics (bit rates, packet sizes, and inter-packet times) [11], and an empirical network traffic analysis of On-Live and Gaikai [12]. However, the QoE to QoS mapping shall be redefined, given the rendering component is resided on the local terminal. Due to multiple remote invoke between components, the impact of network QoS parameters will intensively impact the QoE for players.

### 2.2 Interval Reporting

*Interval reporting* is the most conventional design for status monitoring [5]. The mobile devices set up real-time monitoring on access network performance measurements, including bandwidth, latency, jitter, etc; and user-end device status measurements, including battery capacity and power consumption, network utilization, computing capacity (including the usage percentage of CPU, GPU, memories), the player input and game screen dynamics, etc, and reports these data to the cloud server at intervals. However, to select an optimal interval is still a critical issue for *Interval reporting* approach. Frequently sending detailed monitoring data makes the cloud measurement and prediction accurate, however, it also introduces more network overhead to the gaming system. Moreover, to report all status data to the cloud server is either not a smart move. As a matter of fact, not all of the status need frequently updating. Apparently, the *Interval reporting* is not bandwidth efficiency solution.

### 2.3 Mobile Agent for Information Collection

Mobile agents are composition of software package that is able to migrate from cloud server to user-end devices, and continues its execution on the destination terminals. It provides a potential solution to provide this distributed monitoring, decision and learning capabilities. Mobile agent has been extensively studied in collecting data for wireless sensor networks (WSN) [13], which can address the software (re)installation issue for dynamically changing application functionalities (or their requirements). As a special kind of software, a mobile agent migrates among sensor nodes to carry out task(s) autonomously. For instance, collecting sensory data from a number of source nodes with some processing defined on demand, and adaptively handling sensory data depending on time-varying network dynamics are typical application requirements of the mobile agent dispatcher (i.e., the sink node) in WSNs. Using mobile agent has been shown to

be an efficient approach to enhance such capabilities of WSNs. In this work, we apply mobile agent as a information collector to our environment perception solution for cognitive cloud gaming.

## 3 Design of Environment Perception Solution

A key function of the situation-awareness cognitive platform is the capability to perform real-time environment perception that measures cloud performance, access network performance and end-user device status. For the performance measurement on cloud, the cloud server is capable to monitor the real-time status, including latency inside the cloud (authentication latency, processing latency, internal network latency), and processing capacity consumption and availability, etc. In contrast, access network performance and user-end device status requires a reporting mechanism between the cloud and mobile devices.

### 3.1 Threshold-based Data Reporting

An intuitive data collection solution to improve the efficiency of *Interval Reporting* is *Threshold-based Data Reporting*. It sets up a threshold and reports the requested data when the variation of monitoring data exceeds the predetermined threshold.

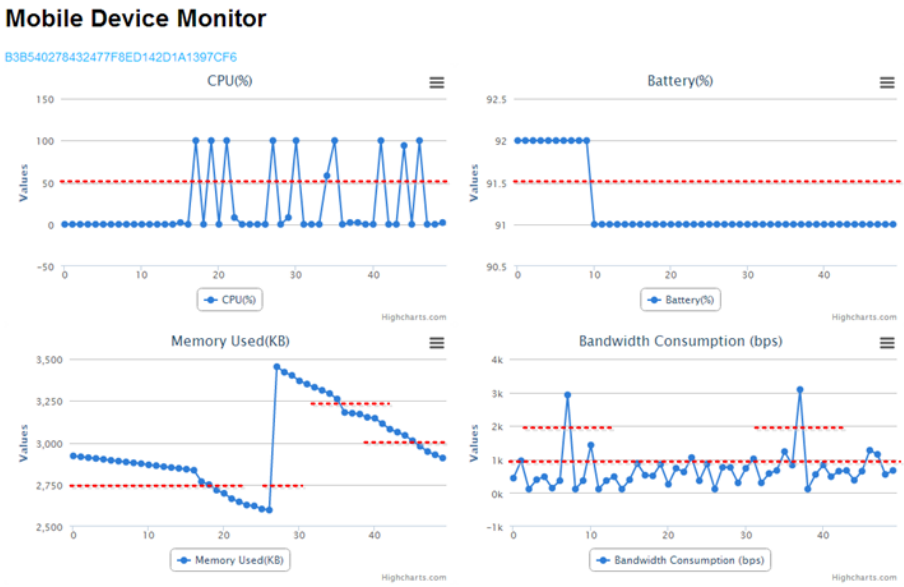


**Fig. 1.** Data Report with Threshold

Fig. 1 depicts an screenshot of conventional interval reporting, in which the mobile devices consequently send the CPU usage, battery percentage, memory usage, and

bandwidth consumption information to the cloud server in the interval of one second. As illustrated in red dot lines, if we introduce a threshold to the existing interval reporting system, the client only update the requested data when the value changes surpass the threshold, thus, the communication frequency is significantly reduced. However, to determine the value of threshold is still a critical issue, since the requirements of environment perception are subject to various of:

- **game genres and scenes**: different game genres and scenes have different QoE criteria, e.g. a First-Person Shooter Game (FPS) may always be more sensitive to network bandwidth than round-based RPG game.
- **monitoring parameters**: distinct monitoring parameters have their own characteristics. For instance, the battery keeps decreasing until the devices are connected to the power, while the memory and CPU percentage are fluctuating along with application usages.
- **terminals**: terminals with various hardware capacities, such as screen resolution and battery, contain unique behaviors from each other.
- **networks**: players are accessing the cloud gaming services through various network with different characteristics, especially when they are moving.

Moreover, compared to continuously reporting through Internet, environment perception procedure can be processed partially in local devices to reduce the communication cost. Therefore, a cognitive, flexible and intelligent environment perception solution is in need.

### 3.2 Mobile Agent Information Collection

To address these issues, we investigate the use of mobile agents [7] dispatched by the cloud to enable these real-time measurements and report the results to the cognitive platform to enable situation-aware adaptations. A task-customized mobile agent migrants from the cloud server to the user-end devices as the cognitive platform required, in order to collect in-need data and report the processed or fused information to the cognitive platform, drawing on the example of mobile agent system for wireless sensor network. The collecting mechanism, including agent design, data processing method, and agent itinerary planning, shall be both latency-sensitive and energy-efficient.

Fig. 2 illustrates the environment perception procedure conducted by the cognitive control center, which is called *cognitive engine* in this context:

1. The *status monitor* of cognitive engine on the mobile devices consecutively read status data from the system kernel. Instead of reporting to the cloud server at intervals, it stores these information in the local status database for future inquiry from the cloud server.
2. In the beginning of the game session, the cognitive engine on the cloud server initiates a mobile agent with inquiry code and dispatched it to the mobile devices via network. The designated mobile agent is interpreted and executed in the *agent executor* on the mobile devices. The process of mobile agent execution, so called *local analysis*, retrieves the unread data stored in the *status database* and performs

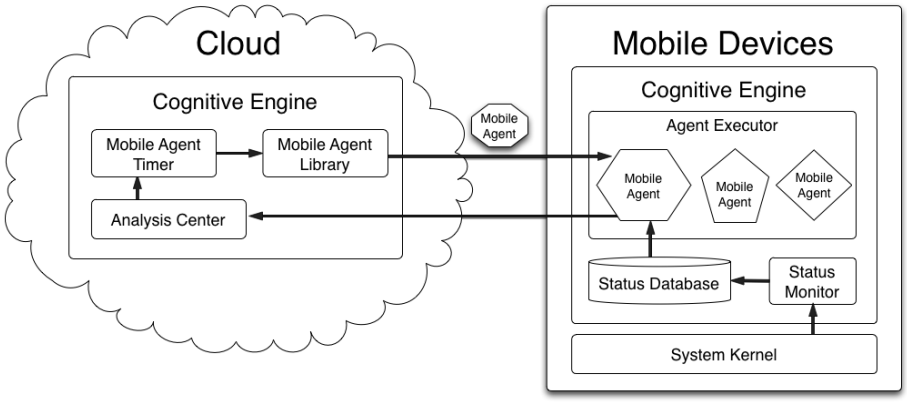


**Fig. 2.** Environment Perception for Cloud Gaming Platform

analysis to evaluate the overall performance of the access network and user-end device status.

3. Once the process of *local analysis* accomplished, the mobile agent then report the results, named *device current status*, to the *analysis center* in the cloud server through network.
4. The *analysis center* on cloud is a decision support system, which conducts the analysis of all status data, including current and historic status of access network, user-end devices status and cloud server performance. The procedure of measurement and prediction is called *cloud analysis*.
5. The *mobile agent timer*, set by the *analysis center*, triggers the *mobile agent library* to dispatch task-customized mobile agent to the mobile devices for next round status information collection.

The mobile agent information collection mechanism provides certain benefits to the cognitive engine for mobile cloud games, including:

– **Intelligent Collecting Interval:** After *cloud analysis* on *analysis center*, the cloud is able to predict the variance of the access network and user-end device status, therefore, determine an optimal collecting timing for next mobile agent.
– **Flexible Agent Design:** Also instructed by the *analysis center*, the cognitive engine in the cloud is able to create most task-specific mobile agent for dispatching, which provide flexibility to fetch the information the cloud most interested in.
– **Efficient Information Transmission:** One of most important feature of mobile agent is to execute locally to perform information retrieval, fusion and collection, which means the network transmission will be more efficient and the network overhead introduced will be reduced. This is especially important, since the cloud-based gaming system is strong network-dependant.

### 3.3 Design of Local Analysis

Since the *local analysis* is performed by mobile agent, the design of local analysis is intrinsically the customization of the mobile agent. As previous discussed, the mobile agent is task-oriented, and the variety of executable code in mobile agent is considered as the most significant feature. However, whatever changes, the principle of local analysis is to concern the accuracy and efficiency of the reporting data.

Here we demonstrate an example for mobile agent design. Given a set of $T = t_i, t_{i+1}, t_{i+2}, ..., t_j$ representing the duration of monitoring time and a set $C$ representing the type of monitoring data, such as bandwidth, latency, jitter of access network and battery capacity, the usage percentage of CPU, GPU, memories of mobile devices. We denote the monitoring status value as $s(t, c)$, representing the $c$ status value at time $t$. A mobile agent code focus on retrieving the raw profile of the monitoring information is designed to report the value of mean $E(c)$ and deviation $D(c)$, defined as following:

$$E(c) = \frac{\sum_{t \in T} s(t, c)}{t_j - t_i} \tag{1}$$

$$D(c) = \frac{\sum_{t \in T} (s(t, c) - E(c))^2}{t_j - t_i} \tag{2}$$

Note that, the flexibility of mobile agent is not only on the functionality of the executable code but also the interval of agent dispatching. To choose the optimal interval and appropriate mobile agent are implemented in *analysis center*, which will be discussed in next section.

### 3.4 Design of Cloud Analysis

The *cloud analysis* on *analysis center* is the core controller of the cognitive engine. It read all reporting information from mobile agents and perform measurement, prediction and joint-optimization for the cloud gaming platform. With the support of all status data, including cloud performance monitoring from the server script, current and historic data of access network, user-end devices status from mobile agents, the procedure of *cloud analysis* is supposed to yield two fundamental results:

- **QoE Level Factor:** The factor is an intelligent prediction from the measurement of overall system performance. It is a quantized value as the reference for the intelligent adaption of the cognitive platform.
- **System Variance Factor:** The factor predicts the variance of the overall system, which determines the time and type for next mobile agent to be dispatched.

## 4 Implementation of Environment Perception

### 4.1 Enabling Technologies

To facilitate the proposed environment perception, we have implement a experimental platform as illustrated in Fig. 3, including cloud server and mobile client.

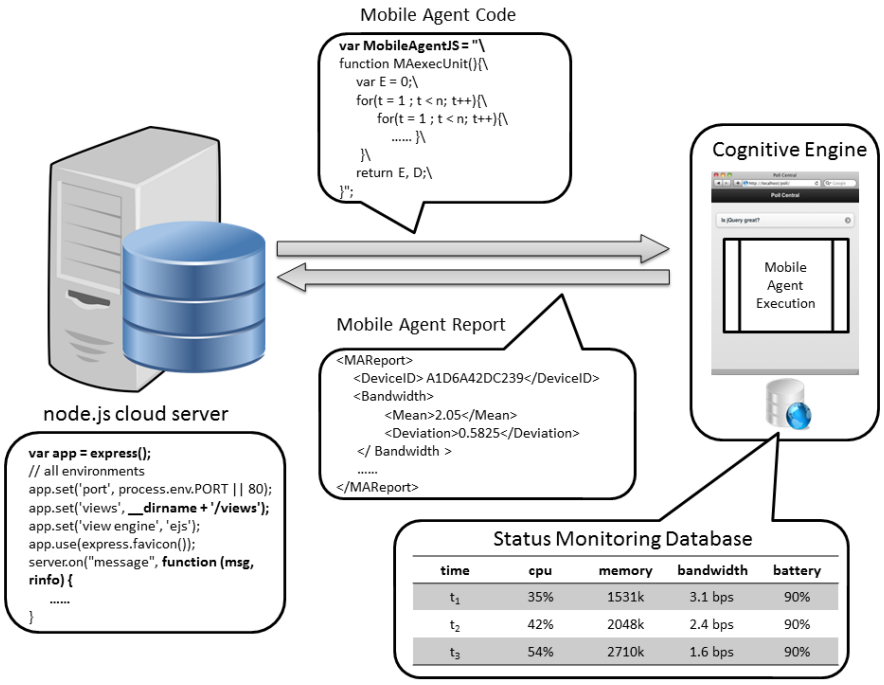


**Fig. 3.** Implementation of Environment Perception

One of the key features of this work is to make the mobile agent migrate and execute in the mobile devices. However, the code generation and compiling in various devices are long-time troubling issues for mobile agent utilization. In this work, we innovatively adopt JavaScript, an interpreted computer programming language, to construct mobile agents. JavaScript was originally implemented as part of web browsers so that client-side scripts could interact with the user, control the browser, communicate asynchronously, and alter the document content that was displayed. More recently, however, it has become common in both game development and the creation of desktop applications.

To make the system in a constant programming style, we deploy node.js as the software system for our cloud server. Node.js is a server-side software system designed for writing scalable Internet applications, notably web servers. Programs are written on the server side in JavaScript, which enables web developers to create an entire web application in JavaScript, both server-side and client-side. This feature will facilitate our future work on mobile agents that can be running both on server and client.

For the mobile client, we embedded a WebKit-based browser to parse and execute the JavaScript mobile agent from the cloud server. In our implementation, the WebKit browser is built on Android smartphone. However, all mobile operating systems supporting browsers are able to support our cognitive platform after a small number of modification.

### 4.2 Stateless Connection

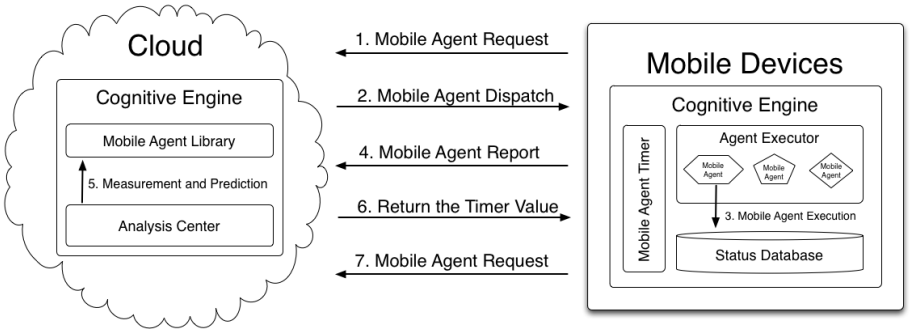


**Fig. 4.** Mobile Agent Paradigm for Environment Perception

Mobile agent dispatching pattern in our implementation is slightly modified to the concept design. Instead of initiative mobile agent push by the cloud server, we set up the mobile agent timer on the mobile devices. When the timer expired, the mobile client post a mobile agent request to the server, and the cloud response the mobile agent code. The reason is that, if the timer is running on the cloud as designed in Section 3, a persistent connection between the cloud and the user-end devices shall be maintained during the whole gaming session, which is not as flexible as stateless connections. However, the mobile agent dispatching interval should always controlled by the *Analysis Center* in the cloud, therefore, the cloud need to response the mobile devices with the value of mobile agent timer, as phase 6 depicted in Fig. 4.

### 4.3 Mobile Agent Cache

Transmitting mobile agents cost bandwidth as well. In our implementation, once a mobile agent has been fetched from the cloud server, it is cached in the *agent executor*. Whenever the client request the same agent, the cognitive engine will check with the cloud to see if the agent code is up-to-date. If no updates is in need, the *agent executor* will run the local version of mobile agent, on the purpose of saving network bandwidth.

## 5 Conclusion

Gaming as a Service has been introduced to the public as next generation entertaining platform. In order to provide a acceptable quality of experience for players in dynamic network context, a gaming platform is in need to provide cognitive capacity of resources and characteristics of the cloud, the access network, and the end-user devices, and enables dynamic utilization of these resources. In this work, we design a mobile agent based paradigm to implement a cognitive environment perception solution with a novel capability to learn about the game player's environment to facilitate the envisioned

cognitive gaming platform. A pure JavaScript solution is adopted to overcome the compiling issues of software migration and creates an efficient information reporting for the cognitive platform.

## Acknowledgement

This work is supported by a University of British Columbia Four Year Doctoral Fellowship and by funding from the Natural Sciences and Engineering Research Council.

## References

1. W. Song and X. Su, "Review of mobile cloud computing," in *Communication Software and Networks (ICCSN), 2011 IEEE 3rd International Conference on*, may 2011, pp. 1 –4.
2. K. Yang, S. Ou, and H. Chen, "On effective offloading services for resource-constrained mobile devices running heavier mobile internet applications," *Communications Magazine, IEEE*, vol. 46, no. 1, pp. 56–63, 2008.
3. R. Shea, J. Liu, E. Ngai, and Y. Cui, "Cloud gaming: architecture and performance," *Network, IEEE*, vol. 27, no. 4, pp. –, 2013.
4. S. Wang and S. Dey, "Modeling and characterizing user experience in a cloud server based mobile gaming approach," in *Global Telecommunications Conference, 2009. GLOBECOM 2009. IEEE*, 30 2009-dec. 4 2009, pp. 1 –7.
5. S. Wang and S. Dey, "Addressing response time and video quality in remote server based internet mobile gaming," in *Wireless Communications and Networking Conference (WCNC), 2010 IEEE*, 2010, pp. 1–6.
6. W. Cai, Conghui Zhou, V. Leung, and M. Chen, "A cognitive platform for mobile cloud gaming," in *2013 IEEE 5th International Conference on Cloud Computing Technology and Science (CloudCom)*, Dec 2013.
7. D. Lange and O. Mitsuru, *Programming and Deploying Java Mobile Agents Aglets*, Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1st edition, 1998.
8. M. Jarschel, D. Schlosser, S. Scheuring, and T. Hossfeld, "An evaluation of qoe in cloud gaming based on subjective tests," in *2011 Fifth International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS)*, June 30-July 2 2011, pp. 330 –335.
9. Y. Lee, K. Chen, H. Su, and C. Lei, "Are all games equally cloud-gaming-friendly? an electromyographic approach," in *Proceedings of IEEE/ACM NetGames 2012*, Oct 2012.
10. S. Choy, B. Wong, G. Simon, and C. Rosenberg, "The brewing storm in cloud gaming: A measurement study on cloud to end-user latency," in *Network and Systems Support for Games (NetGames), 2012 11th Annual Workshop on*, 2012, pp. 1–6.
11. M. Claypool, D. Finkel, A. Grant, and M. Solano, "Thin to win? network performance analysis of the onlive thin client game system," in *Network and Systems Support for Games (NetGames), 2012 11th Annual Workshop on*, 2012, pp. 1–6.
12. M. Manzano, J. Hernandez, M. Uruena, and E. Calle, "An empirical study of cloud gaming," in *Network and Systems Support for Games (NetGames), 2012 11th Annual Workshop on*, 2012, pp. 1–2.
13. W. Cai, M. Chen, T. Hara, L. Shu, and T. Kwon, "A genetic algorithm approach to multi-agent itinerary planning in wireless sensor networks," *Mob. Netw. Appl.*, vol. 16, no. 6, pp. 782–793, Dec. 2011.