Optimized Game Object Selection and Streaming for Mobile Devices

Mahdi Hemmati, Shervin Shirmohammadi, Hesam Rahimi, Ali Asghar Nazari Shirehjini
Distributed and Collaborative Virtual Environments Research Laboratory (DISCOVER Lab)
School of Electrical Engineering & Computer Science, University of Ottawa
Ottawa, Ontario, K1N 6N5, Canada
mhemmati@eecs.uottawa.ca, { shervin | hrahimi | ali.nazari }@discover.uottawa.ca

Abstract — **There has been a recent increase in research efforts to enable real-time delivery of virtual environments (VE) and multiplayer game content to mobile devices in a way that overcomes the limitations of these devices such as longer network delays due to the wireless setting and limited battery life. Previously, we had introduced an approach to tackle the streaming of 3D objects for such environments with the goal of improving the real-time response and interactivity of networked VEs and games, using an activity-aware 3D object selection and prioritization scheme, before streaming those objects over the network to the mobile device [19]. In this paper, we introduce an optimization scheme to optimize object selection and prioritization with respect to globally defined constraints of energy consumption and bandwidth constraints, or qualitative requirements such as immersion and visual quality. Our results show that the optimization scheme leads to the best quality of gaming experience from the user's perspective.**

Keywords- game streaming; activity awareness; object selection and streaming optimization; virtual environments; mobile games

I. INTRODUCTION

There is no doubt that mobile computing is not just on the rise but is truly the future of computing for consumers. Reuters [1] has reported that already more than half of the world population use mobile devices; hence, people are more likely to use a handheld device than a PC to browse the Internet [2]. At the same time, we are witnessing a move towards the mass consumption of 3D media, with a significant rise in applications such as 3D video, 3D virtual environments (VE), and 3D games. Of these, the most prevalent are online video games and online 3D virtual environments, with very impressive statistics [32]: 78.6 million people in the U.S. played mobile games in 2009, and downloads of mobile games increased tenfold compared to 2003 [30], while more than 125 million people played mobile games in the UK and US in 2012 [29]. While gaming in general is growing at a rapid pace, mobile gaming is seeing the fastest growth [31], and ABI Research reports that mobile gaming revenues will grow from US$5 billion in 2011 to more than US$16 billion in 2016 [28]. Indeed, the growth of mobile gaming is happening so quickly that industry observers believe mobile games will change the basis of competition in the gaming industry [30]. Mobile gaming has therefore turned into a hot research topic for both the industry and academia.

Due to the dynamic and large amount of content in 3D virtual environments and games, prior download of such worlds is not a practical solution. As an example, Second Life [5] hosted 270 terabytes of user-generated dynamic contents as of 2009 [6][7]. Moreover, due to the limited capabilities of mobile handheld devices in terms of storage, they cannot have the entire VE installed on the device. In addition to the impracticality of pre-installation, download of these large games/VEs through the Internet is very time consuming and not pragmatic. In VEs and games, the user must be able to view the game content almost immediately and start playing the game as soon as possible, instead of waiting for a progress bar as is common in presentational applications such as audio/video broadcasting. Many online game portals will not accept a game in which some sort of gameplay does not start after downloading at most 1 MB of data [8]. Therefore, online and real-time delivery of 3D content over the network is needed to allow user interaction without waiting for a full download. This process is called *3D Streaming* whereby 3D contents such as 3D objects, textures, animation, and scene graphs are sent to the player just-in-time and as needed. This also allows designers to update the models without rebuilding the whole project; i.e. some 3D contents such as additional weapons, environments, characters, or even full levels that are not originally built into the game can be streamed and added later on.

As mobile handheld devices have limited capabilities and resources such as limited battery, limited bandwidth, higher network latency due to the nature of wireless networks, limited processing capabilities and limited-size display, novel techniques are needed to efficiently use local resources of handhelds to accomplish 3D streaming as optimally as possible. In our previous work, we presented an activity-centric 3D streaming architecture as a solution to make online 3D gaming experience richer and with a higher user interaction rate for mobile handheld devices [19]. We introduced the idea of *prioritized activity-centric progressive streaming*, whereby in each frame of gameplay, we consider the context of the game to decide which objects are more important for the accomplishment of the player's current activity. Therefore, less relevant objects in the scene will not be streamed at all or will be streamed at a lower quality, freeing up the resources for objects that are more relevant. By doing so, we not only reduce the number of objects needed to be streamed from the server to target devices, but we also improve the player's performance in the game. However, one of the issues that remained unresolved was an object selection "threshold" setting that was configured manually and was based on multiple experiments with the specific game to find the best value.

In this paper, we remove that manual step and replace it with an optimization scheme to optimize object selection and prioritization with respect to globally defined constraints of the player's mobile device in terms of energy consumption and bandwidth constraints, or qualitative requirements such as immersion and visual quality. Experiments show that our optimization scheme automatically achieves a level of quality which is higher than the above-mentioned manual process.

II. Related Work

In large virtual environments (VE), a user only interacts with a subset of the environment that is visible to him at a given time. Existing approaches for 3D content delivery take advantage of this fact, and can be classified into two major classes: region-based and interest-based. Region-based approaches only stream the geometry information for the player's specific region. Examples include DIVE [9], CALVIN [10], Spline [11] and VIRTUS [12]. In such systems, the environment is divided into a number of "pre-defined" regions and before the user interacts with a given region, the full content of the region must be downloaded. Therefore, in such systems, whenever the user changes the region, interactivity of the system becomes slow as multiple pauses are needed for a region to be completely downloaded. In contrast to a region-based approach, interest-based techniques use an area of interest (AOI) to determine object visibility and the client only receives update messages within the AOI [13][14]. NetEffect [15] is among such systems. These approaches however do not provide any mechanism to control the visual quality, although they may reduce the amount of game content for downloading. Moreover, the download time might still be too long since in recent high-quality games, there may be too many objects inside the mentioned AOI.

Another shortcoming with existing approaches is that all objects within the region or AOI are treated the same, no matter how important they are for the user's current gaming context. The work in [16] and [17] addresses this issue to some extent by focusing on prioritization of objects in a given scene based on the idea of object scope and viewer scope, where each object in the game is associated with an *object scope*, which is a circular region defining how far the objects can be seen, and each player is associated with a *viewer scope;* and their idea is to transmit only the objects whose scope has some overlap with the player's viewing scope. While this improves the performance, priority is simply defined as whether or not an object's scope is inside the viewer's scope, and does not consider game context. In other words, priority will be given always to objects closer to the player, and the proximity of the player with the objects is the only parameter considered in streaming. We showed in [19] that proximity, distance, or user visibility is not always the best method to select a subset of objects for streaming. While such distance-based approaches provide a better visual quality for closer objects, other objects (far objects) are not transmitted or only streamed at a lower quality. However, the context under which the virtual world is rendered does not depend only on distance between objects. For example, when fighting an enemy in a jungle, the enemy has higher priority than the trees and plants around the player, and should be updated with higher quality in terms of both resolution and update frequency, no matter how close or far the enemy is in the visible range. If distance is used as the only parameter for streaming, many close objects such as nearby trees and bushes are sent to the player at a higher quality than required and without having a semantic relevance for the game context. While this might be acceptable on a personal computer, it leads to a waste of resources on mobile devices. In addition, some relevant objects might be transmitted only at a lower quality because they are far, even though they might be highly important, leading to a low quality of gaming.

In contrast, our approach considers the importance of an object for the current situation of the player within the gaming world. This assures that the most important objects are streamed to the player with higher quality, where importance is defined by the game designer within the game context. Our work in [18] described our preliminary idea, and was later expanded in [19], and also showcased in [20] where we demoed a real example of our system working on a personal computer and on a mobile device simultaneously. In this paper, we add to our system the ability to automatically optimize object selection and prioritization with respect to the globally defined constraints of energy consumption and bandwidth constraints, or qualitative requirements such as immersion and visual quality.

III. Proposed Method

For self-containment purposes, we explain here briefly how our method works, and refer the reader to [18] and [19] for more details. Our approach streams only a small portion of the environment to the client in each game frame. This small portion contains objects that the user is mostly interested in; i.e., objects that are needed to fulfill the current gaming task and keep the game enjoyable. For example, as mentioned, when fighting an enemy in a jungle, the enemy has a higher priority than the trees and surrounding plants, and should be updated with higher quality in terms of both resolution and update frequency. Therefore, in each frame of the gameplay, we find the current activity undertaken by the player, which is determined by an *activity recognizer* component. The design of such component is out of the scope of this work and has been investigated before by other researchers, such as the Hierarchical Hidden Markov Model based method proposed in [21], which could be used to predict future context. Moreover, we can have virtual sensors within the game engine that sense and infer higher level knowledge such as the current activity and other game states. Most of this information in most of the games is apriori known anyway, since most physics engines and AI modules already know in which situation the player is and have full knowledge about player in-game activities, because many game animation sequences or dynamic game level adjustments need this information. So we just need to feed this information into our proposed method. After recognizing

the player's current activity, objects are prioritized based on their importance for that activity. In other words, using normalized importance factors, we set the "priority" property of each object for the current activity. After this prioritization, we sort objects based on importance and we stream them in order of priority to the player's mobile device, in one of the following ways:

- We could stream all objects in the decreasing order of priority. However, this won't work if the player's mobile data plan has a limit, which is typically the case.
- Depending on the type of internet connection in use, we could define a maximum download size, set by the service provider or the mobile client. Our framework then assures that maximum download size will not be violated. This will be done by selecting and streaming an optimum subset of objects from the prioritized list. This subset contains a number of objects such that the sum of total importance of those objects is the maximum compared to all other possible subsets of objects. Also, the total size of those objects must be less than a user-defined constraint. For example, if a player does not want to have more than 10 Megabytes of download while playing the game, our system must ensure no more than 10 Megabytes worth of objects from the prioritized list is sent. This is in addition to the fact that those selected objects would have the maximum total priorities among all possible subsets with a size of less than 10 MB. This optimization mechanism is discussed in section III.B below.
- Also, it would be possible to have more constraints defined by the mobile client. As an example, apart from having a maximum download size, a player might have limitations in battery life of the mobile device, and so the game must be played such that the maximum energy usage does not go beyond a specific threshold.

In summary, based on the specific needs of a given player, we can select an optimized subset of objects from our prioritized list while meeting the limitations and player constraints during gameplay. This is explained next.

A. Object Selection and Prioritization Algorithm

Our algorithm is summarized in Figure 1. In each frame of the game, the current activity (*act*) undertaken by the player is fetched. Then, for a player's current scene in the game, we must extract all the objects in that specific scene. This object extraction should take into account the current game state, including the current activity (*act*) and player and enemy position and orientation. In our pseudo-code, we have shown this as a list named *listOfAvailableObjects*.

Our algorithm then looks for the *importance factors* of those selected objects and sets the *priority index* of each object. Now, we have a sorted list of all required objects for a given activity undertaken in a given scene. Next, we must select a subset of these objects that will be supported by the user constraints such as maximum download size, battery life, etc. In the following section, we will explain our approach to selecting an optimized list of objects based on user constraints.

B. Optimization of Objects Subject to User Constraints

We are seeking to pick and stream a selection of objects in the player's visibility range which are the most important ones for the player's current activities; i.e., we want the subset of objects with a maximum total importance for the current activity of the player. This maximization is subject to constraints on total download size and battery usage, among others. There is a vast and growing literature on power-aware computing [22] that provides various physical, empirical, abstract and mixed models of battery discharge behavior that could be utilized to design and implement a power-efficient computing and communication scheme for a mobile device. However, for the case of object selection and streaming, the most significant factor in both battery and bandwidth usage is the amount of data downloaded by the mobile device. Therefore, it would be reasonable to consider total download size as the main constraint of the object selection problem.

Algorithm 1: Activity-Centric Streaming Algorithm

Input: $batteryLevel$: Available battery at client
Input: $maximumDownloadSize$: Maximum download size limited by the user
Input: $priorityIndex[]$: An array containing normalized importance factors
Input: $objectSizes[]$:An array containing the size of different objects

$act \leftarrow getCurrentActivity();$
$listOfAvailableSceneObjects \leftarrow getSceneObjects\ ();$

// the function *getSceneObjects*() uses the current position
// of the player and enemies. It also checks if an object has
// already been streamed or not.

// get user constraints
$batteryLevel \leftarrow getBatteryLevelFromClient();$
$maximumDownloadSize \leftarrow$
$\quad getMaximumDownloadSizeFromClient();$

$L \leftarrow length\ (listOfAvailableSceneObjects);$
for $i \leftarrow 0$ **to** $L - 1$ **do**
{
$\quad priorityIndex[i] \leftarrow$
$\quad\quad normalized(listOfAvailableSceneObjects[i]);$
$\quad objectSizes[i] \leftarrow$
$\quad getSizeOf\ (listOfAvailbleSceneObjects[i]);$
}

$listOfSelectedObjects \leftarrow$
$\quad optimize(listOfAvailableObjects, priorityIndex,$
$\quad\quad act, objectSizes, maximumDownloadSize);$
$prioritizedListOfObjects \leftarrow$ *Sort listOfSelectedObjects based on their priority index;*
$L \leftarrow lenght\ (prioritizedListOfObjects);$
for $i \leftarrow 1$ **to** $L - 1$ **do**
$\quad$ *Stream* $prioritizedListOfObjects[i]$ *to the client;*

Figure 1: Prioritization and Streaming Algorithm.

We therefore assume that the user will set a total download limit (*TDL*) for the game session. This user input serves as the main constraint of our optimization algorithm, which seeks to maximize the total importance of the selected subset of objects such that the total size of the selected objects do not exceed the download limit determined by the user. Assume that there are *n* objects in the visibility range of the player at a certain scene of the game. If the *priority indices* of these objects for the current activity of the player are denoted by vector $\boldsymbol{p} = [p_1, p_2, \dots, p_n]^T$ and their *download sizes* are $\boldsymbol{s} = [s_1, s_2, \dots, s_n]^T$, then we can formulate the optimization problem as a standard *binary integer programming* problem as follows:

$$\max_{\boldsymbol{x}} \ \boldsymbol{p^T}.\boldsymbol{x}$$
$$s.t. \ \ \boldsymbol{s^T}.\boldsymbol{x} \leq L$$

where $\boldsymbol{x}$ is required to be a binary vector of dimension *n* and *L* is the download limit for the current iteration of object selection algorithm. The above integer programming problem is well known as *knapsack* problem and extensively studied in the literature of combinatorial optimization [23]. Both the objective function and constraints are linear functions of optimization variable $\boldsymbol{x}$ and therefore the problem could be solved using a *linear programming* (LP)-based *branch-and-bound* algorithm [21][24]. Numerically robust and efficient implementations of this algorithm are available as commercial software packages [22][25][26]. In terms of computational complexity, this problem is classified as NP-hard, and in fact the decision version was one of Karp's 21 NP-complete problems. However, there is a pseudo-polynomial time algorithm for solving the knapsack problem [27]. Given the fact that the number of objects considered in each round of object selection is not so large, solving such an optimization problem on the server-side would not cause much computational overhead.

The resulting binary solution vector $\boldsymbol{x}$ determines which objects should be streamed and which ones should not, in order to maintain the download size below the limit while maximizing the total importance of streamed objects. Note that, theoretically speaking, the above optimization problem should be solved for each frame of the game. In practice, the frequency of running the optimization algorithm might be lowered to once for a few numbers of frames.

It must be noted that the download limit denoted by *L* in the following equation is not the same as the *total* download limit entered by the user, which we denote by *TDL*. Each session of gameplay normally consists of numerous iterations of object selection and streaming. While *TDL* is the allowed download throughout the entire gameplay, *L* is the download limit for the current round of optimization problem. Our optimal object selection algorithm includes a budgeting plan to fairly distribute total download limit across the entire level, encompassing several rounds of object streaming. Specifically, assume that *N* is the number of *all* objects in this level of game. Moreover, suppose that I_j denotes the set of indices of those objects in the j^{th} round, which have not been streamed till this round of object selection. A fair value for download limit of the j^{th} round of running the optimization problem is:

$$L_j = \frac{\sum_{i \in I_j} s_i}{\sum_{i=1}^{N} s_i} * TDL \ .$$

The above budgeting plan maintains a uniform distribution of download quota during the entire level.

IV. Evaluations

We applied our optimization scheme to the same evaluation scenario as in [19]. As mentioned before, in [19] we set the download threshold manually and based on repeated trial and error experiments. For this paper, we used the same input data as in [19] and we ran the optimization problem and returned the solution in the form of a bit strings indicating which objects should be selected for streaming for each activity. Using these bit strings, the game was run and the method was evaluated using test cases:

1. A game with prioritization based on distance;
2. A game with prioritization based on our activity-aware model, with manual selection of download threshold;
3. A game with prioritization based on our activity-aware model, with automatic optimization as proposed in this paper.

The metrics we used to conduct our evaluation are:

1. the total amount of data transmitted from the server to the client in each frame of gameplay.
2. Game completion: the time during which the player has managed to hit each of the weapon warehouses, which is an indication of the quality of interactivity and of the game experienced by the player.

As can be seen from the results shown in Figures 2 and 3, our proposed optimization algorithm not only automatically selects objects, but also outperforms the manual trial-and-error approach, and obviously outperforms the distance-based approach, as expected.

V. Conclusion

We enhanced our previous optimization method, which is used for the selection and streaming of gaming objects such that not only the most important objects from the perspective of the player's activity are selected and sent to the mobile device, but also user constraints such as download limit, battery limit, etc. can be met. Our enhancement consisted of an automatic algorithm to determine an optimum object selection threshold, making our method more effective and removing the need for manual selection of this threshold. Our results indicate that our selection algorithm outperforms both traditional approaches (distance based) and our previous manual method.

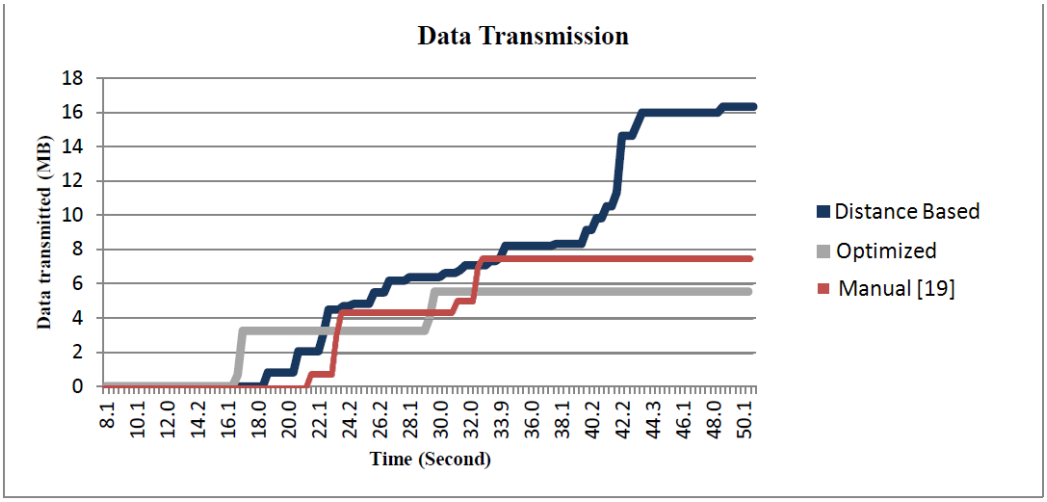

Figure 2: Data Transmission rate

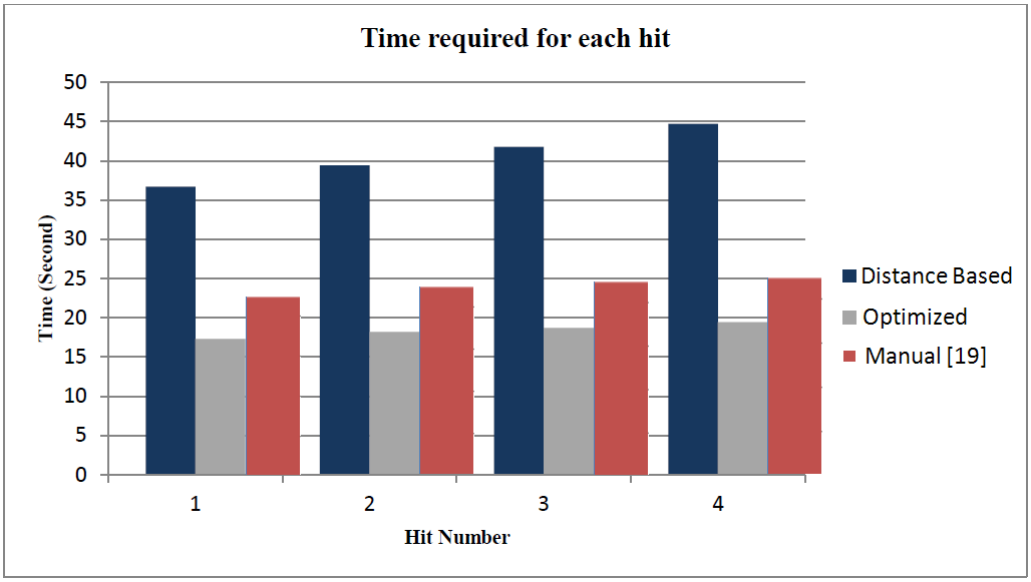

Figure 3: Time required for each hit

REFERENCES

[1] Global Cell Phone Use at 50 Percent (November 29th 2007). [Online]. http://www.reuters.com/article/2007/11/29/us-cellphones-world-idUSL2917209520071129

[2] Mobile Phones Used More than PCs to Browse the Internet (March 31st, 2009). [Online]. http://news.softpedia.com/news/Mobile-Phones-Used-More-than-PCs-to-Browse-the-Internet-108254.shtml

[3] NDP Group, Online Gaming 2008, http://www.npd.com

[4] comeScore Inc. (January 28, 2009). [Online]. http://www.comscore.com

[5] Second Life. [Online]. www.secondlife.com

[6] M-C. Chan, J-R. Jian, C-W. Hung, and W. Tsang Ooi, "Group-Based Peer-to-Peer 3D Streaming Authentication," Proc. Conference on Parallel and Distributed Systems, Shenzhen, China, December 2009.

[7] S-Y. Hu, J-R. Jiang, and B-Y. Chen, "Peer-to-Peer 3D Streaming," IEEE Internet Computing, vol. 14, no. 2, pp. 54-61, March/April 2010.

[8] "Web Player Streaming", Unity Manual, Unity Technologies. [Online]. http://unity3d.com/support/documentation/Manual/Web%20Player%20Streaming.html

[9] O. Hagsan, "Interactive Multiuser VEs in the DIVE System," IEEE Multimedia, vol. 3, no. 1, pp. 30-39, 1996.

[10] J. Leigh, A.E. Johnson, C.A. Vasilakis, and T.A. DeFanti, "Multi-perspective Collaborative Design in Persistent Networked Virtual Environments," in Proc. IEEE VRAIS, 1996, pp. 253-260.

[11] R. Waters et al., "Diamond Park and Spline: A Social Virtual Reality System with 3D Animation, Spoken Interaction, and Runtime Modifiability," Presence, vol. 6, no. 4, pp. 461-480, 1997.

[12] K. Saar, "VIRTUS: A Collaborative Multi-User Platform," in Proc. VRML, 1999, pp. 141-152.

[13] J. S. Falby, M. J. Zyda, D. R. Pratt, and R. L. Mackey, "NPSNET: Hierarchical Data Structures for Real-Time Three-Dimensional Visual Simulation," Computers & Graphics, vol. 17, no. 1, pp. 65-69, 1993.

[14] M. Macedomia, M. J. Zyda, D. R. Pratt, D. P. Brutzman, and P. T. Barham, "Exploiting Reality with Multicast Groups: A Network Architecture for Large-scale Virtual Environments," in Proc. IEEE VRAIS, 1995, pp. 38-45.

[15] T. Das, G. Singh, A. Mitchell, Kumar P., and K. McGhee, "NetEffect: Network Architecture for Large-scale Multi-user Virtual World," in Proc. ACM VRST, 1997, pp. 157-163.

[16] F. W. B. Li, R. W. H. Lau, and D. Kilis, "GameOD: an internet based game-on-demand framework," in Proceedings of the ACM symposium on Virtual reality software and technology (VRST'04), Hong Kong, November 2004, pp. 129 - 136.

[17] F. W. B. Li, R. W. H Lau, D. Kilis, and L. W. F. Li, "Game-on-demand: An online game engine based on geometry streaming," ACM Transactions on Multimedia Computing, Communications, and Applications (TOMCCAP), vol. 7, no. 3, August 2011.

[18] H. Rahimi, A. A. Nazari Shirehjini, and S. Shirmohammadi, "Context-Aware Prioritized Game Streaming," in Proc. of Workshop on Interactive Ambient Intelligence Multimedia Environments, in Proceedings of IEEE International Conference on Multimedia & Expo (ICME 2011), Barcelona, Spain, July 11-15, 2011.

[19] H. Rahimi, A. A. Nazari Shirehjini, and S. Shirmohammadi, "Activity-Centric Streaming of Virtual Environments and Games to Mobile Devices", Proc. IEEE Symposium on Haptic Audio Visual Environments and Games, Qinhuangdao, Hebei, China, October 15 – 16 2011, pp. 45 – 50.

[20] H. Rahimi, A. A. Nazari Shirehjini, and S. Shirmohammadi, "Context-Aware 3D Object Streaming for Mobile Games," in Proceedings of ACM/IEEE Network and Systems Support for Games (NetGames 2011), Ottawa, Ontario, Canada, October 6-7 2011.

[21] B. Hariri, S. Shirmohammadi, and M. R. Pakravan, "Hierarchical HMM Model and Measurement of Online Gaming Traffic Patterns," in Proc. IEEE International Instrumentation and Measurement Technology Conference, Victoria, Canada, May 12-15 2008, pp. 2195 – 2200.

[22] R. Rao, S. Vrudhula, and D. N. Rakhmatov, "Battery Modeling for Energy-Aware System Design," Computer, vol. 36, no. 12, pp. 77-87, December 2003.

[23] S. Martello and P. Toth, Knapsack problems: algorithms and computer implementations.: J. Wiley & Sons., 1990.

[24] M. L. Fisher, "The lagrangian relaxation method for solving integer programming problems," Management science, vol. 27, no. 1, pp. 1-18, January 1981.

[25] K. Aardal and F. Eisenbrand, "Integer Programming, Lattices, and Results in Fixed Dimension," in Discrete Optimization, Handbooks in Operations Research and Management Science 12.: Elsevier, 2005, pp. 171-243.

[26] IBM. (2011, October) IBM ILOG CPLEX Optimizer. [Online]. http://www-01.ibm.com/software/integration/optimization/cplex-optimizer/

[27] M. R. Garey and D. S. Johnson, "Strong NP-completeness: results, motivation, examples, and implications," J. ACM, vol. 25, no. 3, pp. 499-508, July 1978.

[28] "Mobile Gaming", ABI Research Report, July 27 2011, http://www.abiresearch.com/research/1006314

[29] Information Solutions Group, "PopCap Games Mobile Gaming Research – 2012", June 15 2012, http://www.infosolutionsgroup.com/popcapmobile2012.pdf

[30] J. Soh, B. Tan, "Mobile Gaming", Communications of the ACM, 51(3): 35 – 39, March 2008.

[31] N. Bilton, "Video Game Industry Continues Major Growth, Gartner Says", The New York Times, July 5 2011, http://bits.blogs.nytimes.com/2011/07/05/video-game-industry-continues-major-growth-gartner-says/ .

[32] S. Shirmohammadi, M. Hefeeda, W.S. Ooi, and R. Grigoras, "3D Mobile Multimedia", ACM Transactions on Multimedia Computing, Communications, and Applications, Vol. 8, No. 3S, Article 41, September 2012, (3 pages).