Load Balancing of P2P MMORPG Systems with Hierarchical Area-of-Interest Management

Satoshi Fujita
Department of Information Engineering, Hiroshima University
Higashi-Hiroshima, 739-8527, Japan

***Abstract*—This paper studies the load balancing problem in distributed systems designed for massively multiplayer online role-playing games (MMORPGs). More concretely, we consider a distributed system of master-worker type in which each worker is associated with a particular region in the game field, and propose a scheme to balance the load of workers as much as possible. The basic idea of the proposed scheme is to dynamically adjust the number of workers associated with each region according to the number of players in the region. The performance of the scheme is evaluated by simulation. The simulation results indicate that it reduces the response time of a previous scheme proposed by Yu *et al.* by about 50%, which varies depending on the mobility pattern of the players.**

***Index Terms*—Load balancing, master-worker model, MMORPG, area-of-interest.**

I. Introduction

In recent years, online video games such as Final Fantasy XI[1] and EverQuest[2] have attracted many game users. For example, it is reported that the number of active users of Final Fantasy XI reaches one million in 2014[3]. These MMORPGs (massively multiplayer online role-playing games) easily cause a bottleneck at the servers since in these games, a huge number of players interact with each other within a virtual space in realtime. To overcome such a crucial issue, MMORPGs supported by the Peer-to-Peer (P2P) technology have been proposed in the literature [1], [3], [5], [6], [11]. In this paper, we focus on the management of the information on players in such P2P MMORPGs. In particular, we consider the distributed execution of the area-of-interest management (AOIM, of short) [2], [8] in P2P environments.

AOIM is a technique to reduce the management cost of the game server by restricting the *field of vision* of each player (see Figure 1 for illustration). Such a restricted vision field is referred to as the AOI of the player and it generally has a "circular" shape centered at the player. The goal of AOIM is to continuously, selectively notifies game events occurred in the AOI to each player (examples of game events include the movement in the field and the conversation with other players). Since players move as the game proceeds, the system should trace the position of all players to update their AOI, i.e., the set of players contained in the AOI, as well as the tracking of game events related to these players. In order to realize such a management task in a P2P environment with no centralized control, we need to carefully design the way of notifying such events to the players, i.e., who and when notifies the events to the players.

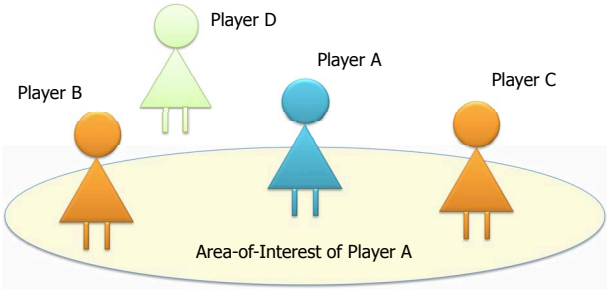


Fig. 1. Area-of-interest of Player A (Players B and C are visible from Player A, but Player D is not).

In this paper, we propose a scheme to realize a quick update of the AOI of players participating in P2P MMORPGs. Our scheme is an improvement of the scheme called MOPAR proposed by Yu *et al.* [10]. As will be described later, MOPAR statically partitions the given game field into several subfields called **cells** and delegates a part of the management task to a peer selected from peers associated to the cell. The selected peer is called the **master** of the cell, and updates the AOI of players in the cell through the communication with masters managing adjacent cells. With such an approach, we could effectively delegate a part of the task of the server to the participants. However, in the original protocol proposed by Yu *et al.*, the number of masters in each cell is fixed to (at most) one, which causes a

[1]http://www.playonline.com/ff11/index.shtml
[2]https://www.everquest.com/
[3]http://www.hd.square-enix.com/eng/pdf/ar_2014_01en.pdf

978-1-4799-8676-7/15/$31.00 copyright 2015 IEEE
SNPD 2015, June 1-3 2015, Takamatsu, Japan

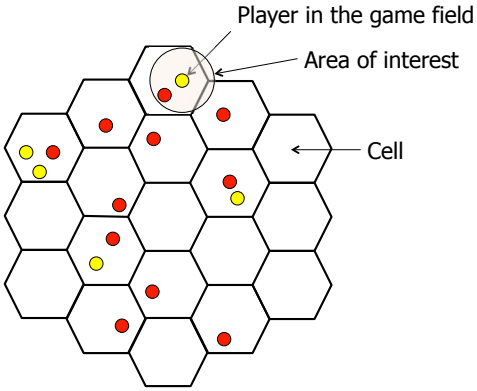


Fig. 2. Game field partitioned into hexagonal cells (yellow circle means a player corresponding to an ordinary peer and red circle means a player corresponding to a master).

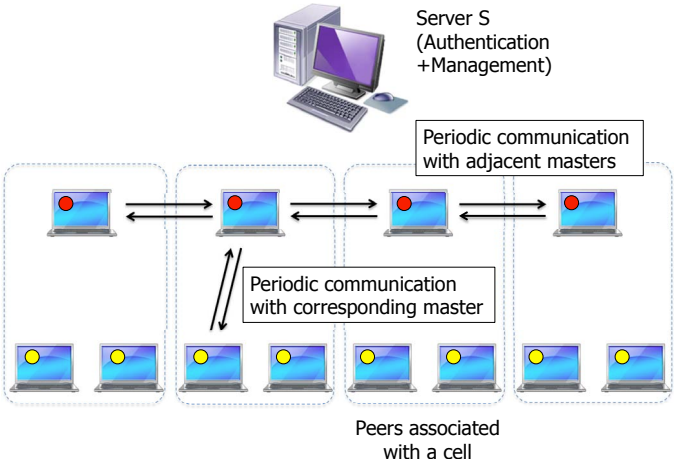


Fig. 3. Communication among peers in the P2P model used in MOPAR (peers with red circle represent masters).

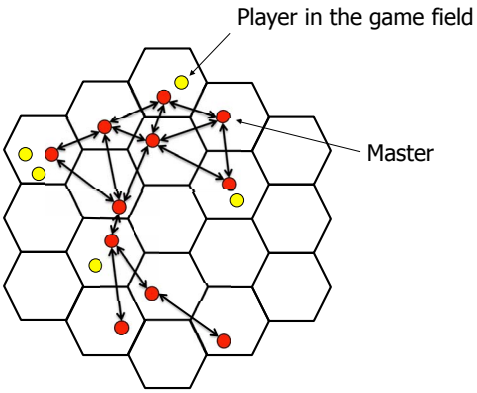


Fig. 4. Communication with masters of adjacent cells.

heavy load if many players concentrate in a specific cell. Such a heavy load of the master increases the response time of the updates, which degrades the comfortability of the resulting game environment. The basic idea of our scheme is to *dynamically adjust* the number of masters in each cell using a threshold-based merge-and-split mechanism. Here, in conducting a splitting, we need to avoid a *discontinuous change* of the AOI such that many players suddenly appears or disappears. In the following, we propose several techniques to resolve such issues. The performance of the proposed scheme is evaluated by simulation. The simulation results indicate that the proposed scheme reduces the response time of the previous scheme by about 50%, which varies depending on the mobility pattern of the players.

The remainder of this paper is organized as follows. Section II describes preliminaries. Section III describes the proposed scheme. Section IV shows the simulation results. Finally, Section V concludes the paper with future work.

II. Preliminaries

A. Model of P2P System

Consider a distributed system consisting of a central server S and a set of peers P, where each peer in P is identified with a player in the game field. The server S plays the following two roles:

- Authentication server which securely manages the charging information such as: 1) the purchase of items and 2) the skill and the level of each player.
- Management server which maintains the set of peers P with their attributes. It does not keep the dynamic information such as game events and the position of players in the game field.

The game field is partitioned into hexagonal cells as shown in Figure 2, where the size of cells can be independent of the AOI of players. Let $\mathcal{R} = \{R_1, R_2, \ldots, R_n\}$ be the resulting set of cells. At each point in time, each player in P *exists* in a cell in $\mathcal{R}$. Let P_i be a variable representing the set of peers existing in R_i.

B. Management of Dynamic Information

For each i with $P_i \neq \emptyset$, the server S selects a peer in P_i according to an appropriate rule and promotes it as the **master** of cell R_i.

The master of cell R_i maintains the information on all peers existing in R_i, such as ID, current position in R_i and IP address, where the position of each player is periodically collected to the corresponding master. The collected information is selectively notified to the relevant peers. More concretely, upon detecting that a player v enters the AOI of player u existing in R_i, the master of R_i notifies the fact to u with the IP address of v so that u can start the direct communication with v

which will (automatically) terminate when the counterpart of communication leaves the AOI. See Figure 3 for illustration. The change of AOI of players is detected by communicating with the masters of adjacent cells; i.e., it could be accurately acquired even if the AOI of a peer intersects with more than one cell as shown in Figure 2. Figure 4 shows the communication between adjacent masters. Note that we can bound the number of adjacent masters to be communicated by three, if the size of cells is sufficiently large compared with the size of AOI.

In MMORPGs, each player walks around the game field to meet other players or to clear specific missions. This causes a frequent crossing of the boundary of adjacent cells. To correctly reflect such a behavior of players to the set of masters, in MOPAR [10], player u conducts the following operations before leaving cell R_i

- If u is the master of R_i, it selects a random peer v in $P_i - \{u\}$ and hands over the role of master to v (such a selection is not conducted if $|P_i| = 1$).
- Otherwise, it simply removes the information on u from the master of R_i.

On the other hand, it conducts the following operations before entering cell R_j

- If $P_j \neq \emptyset$, then u enters R_j by sending its ID and IP address to the master of R_j.
- Otherwise, it simply becomes the master of R_j.

Note that the master of each cell can be identified by referring to the server S (in MOPAR, such a behavior is realized by using the lookup to the Pastry [9]).

C. Model of Peers

Peers are modeled as follows (for simplicity, we assume that the server S has a sufficiently high capability so that it can complete any computation and communication within a unit time). In a unit time, each peer executes the following sequence of operations:

- Receive messages from the input buffer;
- Conduct necessary computation; and
- Send messages to other peers.

Sent messages are delivered to the buffer of receivers within a constant time; i.e., we omit the effect of message routing and congestion avoidance. Each receiver receives a constant number of messages in a unit time, while the size of buffer is not limited to avoid overflow. Unreceived messages should be processed in the next cycle.

III. Proposed Method

A. Classification of Game Events

At first, we classify game events into three types:

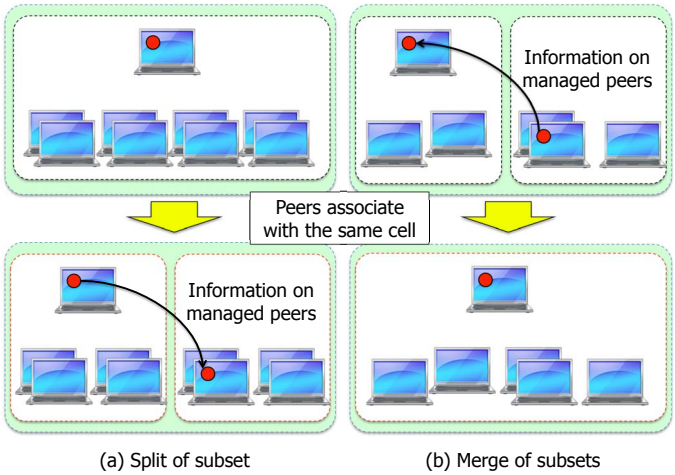


Fig. 5. Split and merge operations.

1) **Type-A:** Events which occur at a specific point in the field, e.g., shopping at a market and boss fight at a specific place (e.g., dungeon or bridge).
2) **Type-B:** Interaction with other players, e.g., conversation and fight, which can start after receiving the IP address on the corresponding peer.
3) **Type-C:** Change of AOI, e.g., the entering and the leave of a player.

The objective of the proposed scheme is to support Type-C events, where in the following, we assume that Type-A events are handled by the server S and Type-B events are realized by the local communication between peers. The reader should note that the support of Type-C events is crucial to realize an attractive online video game since the precise recognition of AOI is a key issue to conduct a timely interaction with nearby players.

B. Load Balancing Scheme

In MOPAR, the number of messages handled by the master of cell R_i is proportional to $|P_i|$ and the load of the master is proportional to $|P_i|^2$ since it needs to calculate the distance for all pairs of the players. Under the model of P2P systems described in Section II, such a load of masters directly affects the update time of AOI. The key idea of the proposed scheme is to dynamically adjust the number of peers managed by each master. More concretely, we associate several peers (masters) to each cell and each master associated with cell R_i manages a subset of P_i.

Let $U(u)$ denote the set of peers managed by master u. Set $U(u)$ dynamically changes by conducting split and merge operations triggered by two thresholds T_U and T_L. More concretely,

- if $|U(u)| > T_U$, u equally partitions $U(u)$ into two subsets and hands over one subset to a new master v

which is randomly selected from $U(u)$ (see Figure 5 (a) for illustration).
- if $|U(u)| < T_L$, then after identifying other master v associated with the same cell R_i (if any), u asks v to merge $U(u)$ and $U(v)$ into a single set (see Figure 5 (b) for illustration).

If $|U(u) \cap U(v)| > T_U$, then the resulting set is split into two halves again, so that the number of peers managed by each master is always within the range of $[T_L, T_U]$. In order to avoid unnecessary fluctuation, we fix two thresholds to satisfy relation $2T_L < T_U$, e.g., $T_U = 3 \times T_L$. Recall that the list of masters is maintained by the server S and is updated appropriately.

By applying split and merge operations, the number of "observable" peers in the AOI changes as follows[4]:

- A split reduces the number of peers to a half, while it is still large enough since the size after the split is at least T_L.
- The number of peers increases by a merge, but it is not too large since the size of the set after the merge (and a subsequent split) is at most T_U.

C. Splitting Algorithms

This paper proposes three splitting algorithms. The first algorithm splits the given subset $U(u)$ by the peer ID, the second algorithm splits $U(u)$ in a random manner and the third algorithm splits $U(u)$ so that pairs of peers which have recently interacted belong to the same subset as much as possible.

1) ID-based splitting: In the first algorithm, which will be referred to as the ID-based splitting, the master u simply divides $U(u)$ into two halves after sorting it in a non-decreasing order of the peer ID. If peer IDs are given in the order of participation, it naturally realizes a situation in which two peers which start the game almost at the same time will be visible with each other even when the game field becomes congested.

2) Randomized Splitting: In the second algorithm, the master u randomly selects $|U(u)|/2$ peers from set $U(u)$ and separates them as a new subset. In other words, for each peer in $U(u)$, a half of peers in the AOI disappear after splitting, independent of the participation order and the frequency of interaction.

3) Cut-Based Splitting: The third algorithm is based on the bipartition of a player interaction graph (PIG, for short) representing the interaction between players. PIG for the master u has vertex set $U(u)$ and an edge set $E(u)$ so that two vertices are connected by an edge if and only if their corresponding players have interacted within the past τ time units (note that u can construct such a graph by collecting all game events from players in $U(u)$). The graph bipartition problem is the problem of partitioning the vertex set so that the size of each subset is (almost) equal and the number of edges connecting two subsets is the smallest. Although the graph bipartition problem is NP-hard, there are several efficient local search algorithms such as Kernighan-Lin (KL) algorithm [4] which solve the problem in a heuristic manner. In the proposed scheme, we use the KL algorithm to realize a split of $U(u)$ in a heuristic manner.

[4]The reader should note that once the IP address of a peer such as the partner in a party is known, it can communicate with the peer even after being disappeared from the AOI due to split operation.

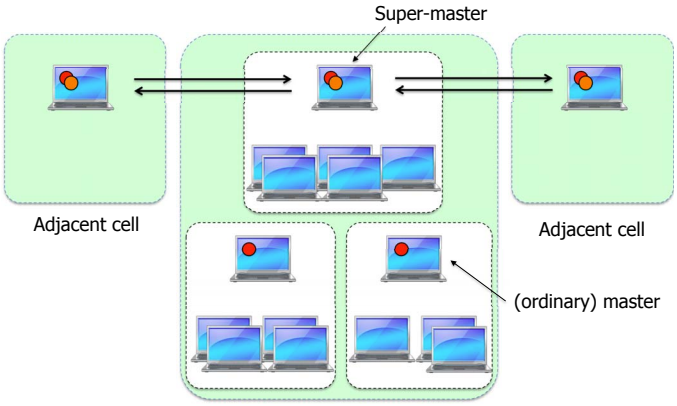


Fig. 6. Proposed hierarchical P2P architecture (ordinary master is marked with red circle and sups-master is marked with orange circle).

D. Efficient Communication with Adjacent Cells

In MOPAR, the master of cell R_i periodically exchanges the position of players in P_i with masters of adjacent cells so that the AOI of each player is correctly updated. In the proposed scheme, we associate several masters for each cell and the number of masters associated with R_i increases as the number of players in P_i increases. This means that we need to take care of the tradeoff between traffic and the freshness of the update of AOI.

Let M_i be the set of masters associated with cell R_i (note that $M_i \neq \emptyset$ as long as $P_i \neq \emptyset$). In the proposed scheme, we promote arbitrary peer in M_i as the **super-master** and use it as a "contact point" for the communication with adjacent cells, to keep the load of each master sufficiently low. Figure 6 illustrates an overview of the proposed hierarchical P2P architecture, where ordinary master is marked with red circle and super-master is marked with orange circle. Let $s(u)$ denote the super-master associate with master u. The concrete procedure for the communication in the proposed hierarchical architecture is as follows:

Step 1: u periodically notifies the list of current positions of the players in $U(u)$ to super-master $s(u)$.

Step 2: $s(u)$ combines messages received from the associated masters including itself into a single message and periodically exchanges it with super-masters of adjacent cells (note that $s(u)$ does not "manage" other masters in the same cell, i.e., it simply acts as a contact point to adjacent cells).

Step 3: Upon receiving a combined message from adjacent super-masters, $s(u)$ forwards it to all masters associated with it. With such a scheme, every master can acquire the information on adjacent cells without directly communicating with the corresponding masters.

Step 4: After receiving the combined message, each master u updates the AOI of the players in $U(u)$, and notifies the update to the corresponding players, if necessary. If the number of peers in an adjacent cell exceeds a certain threshold, u conducts such a calculation merely on players which are close to the boundary of the cell associated with u.

The effectiveness of the above notification scheme is summarized as follows:

- By delegating the role of communication to the super-master, the load of ordinary masters reduces;
- By combining messages received from several masters into a single message, the load of super-master also reduces while it causes a delay in the notification to adjacent cells; and
- By limiting the number of players managed by each master by T_U and by limiting the number of visible peers in adjacent cells by a constant, the load of masters concerned with AOI management reduces.

Finally, to reduce the load of masters concerned with the management of players across boundaries, the proposed scheme conducts a lazy judgement described as follows. When a player v enters cell R_j, it selects a master u associated with R_j according to the rule described in Section III-C and becomes a "tentative" member of subset $U(u)$. It then becomes a "normal" member if it arrives at an inside point of R_j at distance δ from the boundary.

IV. Evaluation

A. Setup

The performance of the proposed scheme is evaluated by simulation using PeerSim [7]. In the simulation, we measure the response time of masters concerned with the change of AOI; i.e., Type C events in Section III-A. The number of players is fixed in the range from 100 to 700 and the game field is divided into 10×10 hexagonal cells, where we assume that the game field is topologically equivalent to the surface of a ball. The length of each side of the hexagon is 100 unit length and the AOI of each player is a circle of radius 50 unit length. In a unit time, each peer conducts the following sequential steps:

1) Move to a point in the game field according to the mobility rules described in Section IV-B;
2) Receive (at most) 50 messages in the input buffer;
3) Conduct necessary computation; and
4) Send messages to other players.

More concretely, in Step (4), each peer notifies its current position to all players in its AOI and the corresponding master. In addition to that, peers promoted as a master or a super-master periodically issues combined messages as was described in Section III-D. Parameters used in the experiments are summarized in Table I.

TABLE I
PARAMETERS.

Parameter	value
Number of players	100 to 700
Game field	10×10 hexagonal cells
Cell	Hexagon with side length 100
Area-of-Interest	Circle of radius 50
(T_U, T_L)	(12,5)
α	0, 50, 100

B. Mobility Patterns

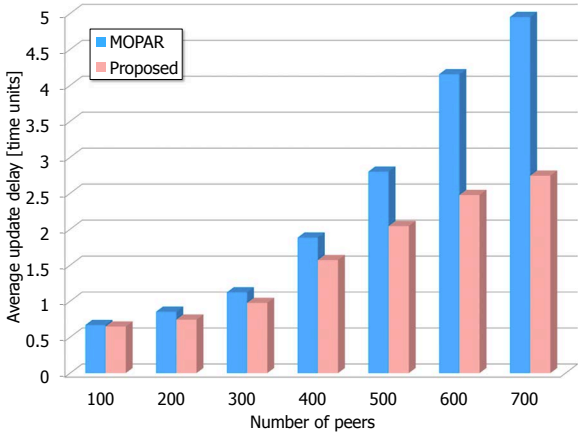
As for the model of mobility of players, we consider two mobility modes called Random and Straight.

In Random, each player conducts random walk. More concretely, in each step, each player follows the movement in the last step with probability 90% and with probability 10%, it moves to a point at distance at most two [unit length] in a random direction. In Straight, each player moves to a specific cell from the initial (random) position with a speed of at most two [unit length per unit time] and after reaching the target cell, it conduct a random walk within the cell until the end of simulation.

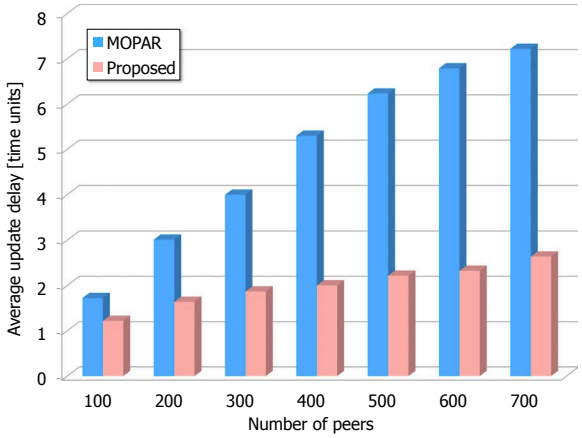
In the experiments, we assume that α [%] of peers follow Random and $100 - \alpha$ [%] peers follow Straight provided that the initial configuration of the peers is randomly given.

C. Result

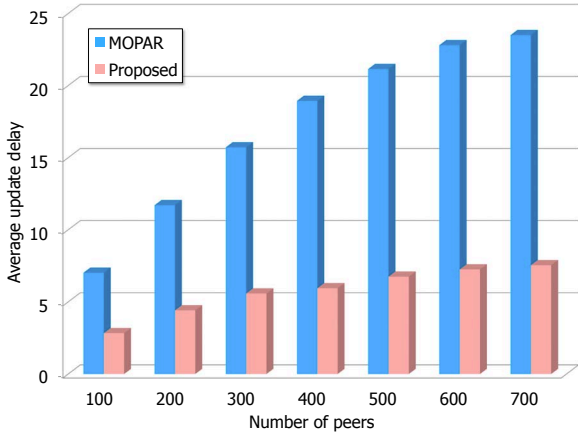
Figure 7 summarizes the simulation results. When $\alpha = 100$, the proposed scheme exhibits almost the same performance with MOPAR for small number of peers, since it does not cause the split of subsets. However, as the number of peers increases, it reduces the response time of MOPAR by about 40% on average. When $\alpha = 0$, on the other hand, the proposed scheme



(a) $\alpha = 100$ [%].



(b) $\alpha = 50$ [%].



(c) $\alpha = 0$ [%].

Fig. 7. Result of simulations.

significantly improves MOPAR, which is apparently due to the concentration of players to a specific cell. In fact, such a concentration degrades the performance of MOPAR and the split of the subset associated with the target cell effectively reduces the average response time. More specifically, the response time of the proposed scheme is bounded by seven [time units] regardless of the number of peers, and it improves the response time of MOPAR by 70% when the number of peers is 700. Similar phenomena could be observed for $\alpha = 50$. Although the amount of reduction is smaller than the case of $\alpha = 0$, it reduces the response time of MOPAR by about 60% on average.

V. Concluding Remarks

In this paper, we propose a threshold-based load balancing scheme for the distributed AOI management in P2P MMPOPGs. The proposed scheme is designed for three-tier P2Ps consisting of ordinary peers, masters and super-masters, and the split of a subset of peers is conducted by solving the graph bipartition problem using KL algorithm. Simulation results indicate that the proposed load balancing scheme reduces the response time of MOPAR by about 50%.

A future work is to conduct extensive simulations to evaluate: 1) the continuity of the change of AOI due to split-and-merge operations; 2) the impact of two thresholds to the performance; and 3) the response time of the proposed scheme in actual distributed environment. Another issue is to enhance the churn tolerance of the proposed scheme.

References

[1] E. Carlini, L. Ricci and M. Coppola. "Reducing server load in MMOG via P2P gossip." In *Proc. 11th NetGames*, 2012, pages 1–2.

[2] S.-Y. Hu, J.-F. Chen and T.-H. Chen. "VON: A Scalable Network for Virtual Environments," *IEEE Network*, 20(4): 22–31, 2006.

[3] T. Iimura, H. Hazeyama and Y. Kadobayashi. "Zoned federation of game servers: A peer-to-peer approach to scalable multi-player online games." In *Proc. the 3rd ACM NetGames*, 2004, pages 116–120.

[4] B. W. Kernighan and S. Lin. "An efficient heuristic procedure for partitioning graphs." *Bell Systems Technical Journal*, 49(2): 291-307, 1970.

[5] B. Knutsson, H. Lu, W. Xu and B. Hopkins. "Peer-to-peer support for massively multiplayer games." In *Proc. IEEE INFOCOM*, Vol. 1, 2004.

[6] M. Merabti and A. El Rhalibi. "Peer-to-peer architecture and protocol for a massively multiplayer online game." In *Proc. IEEE GlobeCom Workshops*, 2004, pages 519–528.

[7] A. Montresor and M. Jelasity. "Peersim: A Scalable P2P simulator." In *Proc. of the 9th Int. Conf. on Peer-to-Peer (P2P' 09)*, 2009, pages 99–100.

[8] A. El Rhalibi and M. Merabiti. "Interest management and scalability issues in P2P MMOG." In *Proc. the 3rd IEEE Consumer Communications and Networking Conf.*, 2006, pages 1188–1192.

[9] A. Rowstron and P. Druschel. "Pastry: Scalable, distributed object location and routing for large scale peer-to-peer systems." In *Proc. of IFIP/ACM Middleware*, 2001, pages 329–350.

[10] A. Yu and S. T. Vuong. "MOPAR: A mobile peer-to-peer overlay architecture for interest management of massively multiplayer online games," *Proc. of NOSSDAV*, 2005, pages 99–104.

[11] A. Yu and S. T. Vuong. "A DHT-based hierarchical overlay for Peer-to-Peer MMOGs over MANETs." In *Proc. 7th IWCMC*, 2011, pages 1475–1480.