COMPUTER ANIMATION AND VIRTUAL WORLDS
Comp. Anim. Virtual Worlds (2015)
Published online in Wiley Online Library (wileyonlinelibrary.com). DOI: 10.1002/cav.1670

SPECIAL ISSUE PAPER

Interest-driven avatar neighbor-organizing for P2P transmission in distributed virtual worlds

Mingfei Wang, Jinyuan Jia, Ning Xie* and Chenxi Zhang

School of Software Engineering, Tongji University, No. 4800, Caoan Road, 201804, Shanghai, China

ABSTRACT

The neighbor table/distributed hash table (DHT) is used to choose the data supplier for data-dispatching services in distributed virtual environments based on peer-to-peer networks. It is essential that a stable and efficient neighbor table/DHT be maintained. Because the avatar has much freedom to roam, the spatial distribution of nodes is not uniform, and the logical topology may change dramatically. Therefore, traditional construction mechanisms, such as the neighbor-discovery mechanism based on spatial distance or DHT, may involve fierce churn in the neighbor table and frequent message exchanges. In this paper, we proposed a dynamic node-organizing mechanism that aims to solve these challenging problems by applying the avatar's behavioral characteristics to the neighbor maintenance mechanism and scene data transmission. First, we have summarized the common social behaviors of avatars and extracted their characteristics. We then propose an interest-similarity measuring algorithm to divide the node into diverse clusters. Next, we measure the cluster stability in terms of interest entropy while constructing a stable neighbor mesh for each node in a cluster. We have conducted extensive simulation experiments that simulate avatar behaviors in a popular massively multiplayer online game. The results show that our proposed mechanism achieved a substantial alleviation of neighbor churn and reduced information exchange, which improves the transmission efficiency in distributed virtual environments. Copyright © 2015 John Wiley & Sons, Ltd.

KEYWORDS

DVE; P2P transmission; avatar behavior; neighbors mesh

***Correspondence**

Ning Xie, School of Software Engineering, Tongji University, No. 4800, Canan Road, 201804, Shanghai, China.
E-mail: seanxiening@gmail.com

1. INTRODUCTION

Currently, 3D virtual reality applications, such as virtual cities [1], industrial simulations, and online games, are spreading widely across the Internet. 3D technology increases the users interactive immersion but brings the challenge of processing data efficiently for huge virtual scenes. For example, the scene data for the popular third-generation 3D role-playing game Ancient Century exceeds 20 GB after installation [2]. The virtual scene area for Second Life, one of the most popular virtual social worlds, has reached 72–109 km^2, with scene data exceeding 100 TB [3]. Therefore, in the real world, the mismatch between limited bandwidth and the real-time download requirements for huge scenes can be substantial, which remains the bottleneck roaming in online virtual worlds.

A hot research topic in recent years is to apply peer-to-peer (P2P) technology to the transmission of virtual scenes to improve scene transmission via node cooperation [4,5]. However, because of the avatars unpredictable roaming behavior in the virtual world, there may be both frequent changes in the logical topology of nodes and severe churn in a nodes neighbor table [6]. The nodes will then need to exchange messages frequently to determine the neighbor relationship, and this will cause heavy network loads and long transmission latencies [7]. Currently, whether in structured P2P networks such as SimMud [8], Colyseus [9], and prediction binary tree (PBT) [10], or in unstructured P2P networks such as pSense [11] and Voronoi-based overlay network [12], even in hybrid P2P networks, a node's frequent moves, joining, and leaving will all lead to many changes in neighbors, which makes it very difficult to maintain the neighbor table/distributed hash table (DHT). Mitigating the impact of the churn in the neighbor table is an urgent problem for the transmission of large-scale virtual scenes.

A virtual world is a simulation and extension of the real world. Similar to the real world, there are hot regions and non-hot regions in virtual worlds [13], and avatars have social characteristics like real-world people, with their own interests and sets of social behaviors. Each avatar will have a special course of action that matches its own interests.

Copyright © 2015 John Wiley & Sons, Ltd.

The members of a group may gather in a particular region because of a common purpose at some moment [14]. These situations occur frequently in current distributed virtual environments (DVEs) [15]. In addition, the interest relationship between avatars that is built in the virtual world can reflect the logical neighbor relationship in the overlay network. Motivated by this observation, we utilize the avatar's interests to help with the efficient transmission of virtual scenes. We propose a dynamic neighbor-organizing mechanism based on avatar interest. According to the avatar's behavior characteristics, we use an interest-similarity algorithm to divide the nodes into diverse clusters, where the nodes in the cluster will form a stable neighbor mesh, and each node may become a neighbor to other nodes. This mechanism addresses the churning issue for neighbor tables effectively, which reduces the message exchange frequency between nodes and shortens the transmission latency.

2. RELATED WORK

2.1. Avatar's Behavior Analysis

Each avatar in the virtual world is mapped to a physical node in the physical network, so the spatial distribution, dynamic behavior, and other behavioral characteristics of the avatar will directly affect the network-organizing mechanism and communication performance of physical network, especially P2P networks [16]. It is necessary to analyze the behavior of the avatar in order to ensure the performance in the virtual worlds.

To neighbor-organizing mechanism in the P2P-based DVEs, there are four most influential characteristics as follows:

(A) Spatial distribution. Because of diverse heat of zone and user's interests, the distribution of avatars is not uniform in the virtual world. The numbers of avatars and the zones appears with power–law relationship [15,17]. The number of zone that the avatar paused follows the long tail distribution [13].

(B) Movement. Avatars roaming have their own destination and follow certain behavioral laws, such as random way point model; avatars move independently between the waypoints that are uniformly and randomly placed. Alternatively, avatar's movement in a group forms reference point group mobility model [18]. In practical, in Second Life, the avatar has three movement states: halted state, exploring state, and traveling state. Usually, they move slowly and chaotically in the hotspots, while fast and predictable in the desert [19]. Moreover, avatars spend most of their time to travel in-between and around a few preferred hotspots [13,15,20].

(C) Grouping. Avatars form groups because of friendship and common interests to achieve same task or reach same destination. For example, in World of Warcraft (WoW), avatars may fight with groups in the battleground [20]. In Second Lift, over 50% of the avatars form groups gather in same region [21].

(D) Pause time. One of the most essential characters is pause time that avatar often stops at a specific place during moving. It follows long-tail distribution. In WoW, around 80% pause time is shorter than 30 seconds, but 100 seconds in Second Life [13]. The pause duration is a perfect occasion for preloading the predicted scene data and supplying service for other nodes [22].

2.2. Neighbor-organizing Mechanisms

According to the relevance of storage content and network topology, P2P networks can divide into structured network, unstructured networks, and hybrid networks.

In structured networks, resources have strict associations with storage locations, commonly defined in terms of a hash table. A structured DVE executes resource transmission and status updates via a DHT. SimMud [8] adopts Pastry and Scribe for data distribution and status updates, respectively. Virtual scenes are divided into fixed-size regions, and the players in the region can only retrieve data for this region. Each region is assigned a region ID and is mapped to a coordinator node to manage each object in the region, the coordinator nodes being organized by Pastry. Each node and resource object is also assigned a unique ID. Using the resources ID, requesters can discover whether the physical node has the requested data. Because of the discontinuities in the scene segmentation, SimMud cannot guarantee that players have a coherent view. If the scope of the region is outside a player's area of interest (AOI), the player will receive irrelevant information. Moreover, another limitation of SimMud occurs when a large number of nodes are concentrated in the same region, leading to the coordinator node becoming a system bottleneck.

The PBT [10] first partitions the scene into many static regions and then utilizes DHT and multicasting to manage the nodes and distribute data, respectively. The nodes obtain the regions information via the region master and release state updates via Scribe. If the nodes are dense in a static region, it can be partitioned dynamically into sub-regions, which become the leaf nodes of the static region. However, the frequent movement of nodes will make it difficult to maintain the PBT.

In unstructured networks, a resource does not have a direct relationship with the storage locations, and there is no global mechanism such as DHT to manage a nodes joining, moving, and leaving behaviors. The nodes mainly depend on a mutual notification system [8] to detect neighboring nodes and to send updated information to their neighbors. In the neighbor-discovery mechanism of pSense [11], a node can forecast the motions of its neighbors based on their behavior. If it has predicted that a neighbor will enter another neighbor's AOI, the node can supply this information in advance, which enables neighbors to establish contact with each other. Voronoi-based overlay

network [12] adopts a scene partition mechanism based on the Voronoi diagram and classifies neighbors as enclosed or boundary neighbors to enable discovery of new neighbors by maintaining updates of the neighbors' table via message exchanges between neighbors, but the neighbor discovery mechanism only depends on the spatial distance of nodes.

A hybrid P2P network simultaneously absorbs the advantages of a structured network (such as centralized management) and an unstructured network (such as scalability). In general, a super peer will be present in the hybrid P2P network and is responsible for global management with all the nodes in its region, which would include interest management, consistency of dynamic information, and security services. One such system [23] divides the whole scene space into several regions controlled by coordinators and separately uses Pastry and Scribe for scene dispatch and state updates. Another system [24] adopts the structured DHT mechanism for state data update, and the unstructured Voronoi mechanism for neighbor discovery, but this is only an initial design and no experimental evaluation. A third system [25] constructs a hybrid architecture via P2P Cloud and uses Dynamic MapReduce and DHT P2P Storage Cloud for dynamic load management. Although these approaches have their advantages, there will be cases where these advantages do not apply.

Furthermore, some works have combined avatar's behaviors analysis with building P2P overlay. Aiming at solving the issue about the beyond load of the frequent state messages updating when the players aggregate around some hotspots, Varvello *et al*. [26] proposed a dynamic clustering algorithm to organize the peers that beyond its maintenance cost based on Delaunay network. Carlini *et al*. [18] proposed the extensive evaluation of an AOI-cast mechanism by considering different mobility models derived from WoW and Second Life. However, there is still no research on applying avatar's behavior analysis to the neighbor-organizing mechanism yet.

According to the survey of avatar behavior in the aforementioned section, we can learn that quantity of players is numerous in the hotspots of virtual worlds, so potential resource-supplying nodes for data request are adequate. However, a large number of neighbor nodes may cause sharply increasing of message exchanges so as to meet the subsequently aggravate network load. Moreover, avatar behaviors such as moving and turning frequently lead the changes of the network including the logical topology transform of the nodes, the scene in the AOI, and the neighbor relationship of nodes. In addition, scene segmentation, without the analysis on scene characteristics and avatar behaviors, will cause avatar frequently switches between scene regions, correspondingly affect the stability of DHT. Alternatively, in the non-hot regions, few avatars lead much fewer neighbors of nodes, even no neighbors, which may increase the server requesting rate even the load of server [27,28].

3. DYNAMIC NEIGHBOR-ORGANIZING MECHANISM BASED ON AVATAR INTEREST

3.1. Main Idea of Our Mechanism

To address the problems of neighbor table churn, frequent message exchange, and no neighbors that occur in current DVE transmission mechanisms, we propose a new neighbor-organizing strategy based on hybrid P2P network that combines avatar behavioral interests with scene data transmission in the virtual world. We divide the nodes into diverse clusters based on spatial distance, movement trajectory, interest preferences, group relationships, and other factors. Nodes in the same cluster tend to maintain the neighbor relationship for substantial periods, and the cluster can update continuously over time. In this way, we can construct a relatively stable neighbor node topology to improve the efficiency of transmission of large-scale virtual scenes.

Figure 1 is a schematic diagram of our neighbor–node organization mechanism. First, we need to preprocess the virtual scene by analyzing the region attraction, partitioning scene and setting the query points in both hot and non-hot regions depending on their specific scene content. The super nodes will collect the node behaviors and extract the node movement characteristics, then compute the similarity between nodes and query points using the interest-similarity algorithm. Next, divide the nodes that have high similarity as clusters and update the cluster dynamically when the avatars roam. Finally, each node chooses nodes from its neighbor mesh from its cluster based on scene data reserved in their memory. Through the aforementioned steps, a topology network will be constructed.

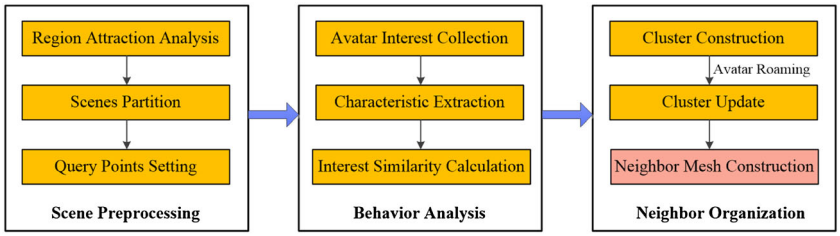


Figure 1. Procedures for neighbor-node organization based on avatar interest.

3.2. Scene Preprocessing

The scene designer needs to set hot and non-hot regions after the analysis of the scene content and the predicted behavioral logic and partition the scene on the basis of the mount of regional scene data and the predicted number of users (nodes), to ensure the overlay network meets the basic stability and load balance [22]. We set query points in models with high attention degree in the scene. Meanwhile, we enable to use the historical data of the popular virtual world based on client/server architecture and empirical distribution to carry out the aforementioned work [13]. More important, for the large-scale scene, the operations such as lightweight and streaming are necessary.

3.3. Interest Characteristic

By analyzing the behavioral characteristics of the nodes in the virtual world, we found two characteristics that play important roles in the virtual world: (i) the group relationship and (ii) avatar trajectories; they have included three factors, that is, spatial distance, moving model, and grouping. We consider these two characteristics in calculating the similarity of node interests and partitioning them into diverse clusters.

Definition 1 (Avatar Group)**.** The nodes in a virtual world will compose a logical group based on the friend relationship, the battle relationship, and the interest relationship at time T_i. We set

$$Group(T_i) = \{P_i \mid P_i \in (R_i, T_i)\}\ (1 \le i \le n) \quad (1)$$

where R_i represents the relationship at time T_i, its value being 0 or 1. If the nodes belong to a group, its value is 1. Otherwise, it is 0.

Definition 2 (Query Point)**.** This is a spatial location that is mapped to a scene object and is selected from the virtual scenes, which could represent the characteristics of a regional scene, such as a famous building or the only path leading to a castle. We set

$$Q = \{q_1, q_2, \ldots q_i, \ldots, q_n\}, (1 \le i \le n) \quad (2)$$

where q_i means the spatial location.

Definition 3 (Spatiotemporal Trajectory)**.** This is a sequence composed of several multidimensional spatial locations in a time series, which has both spatial and temporal properties. We set

$$TR = \{< Location_1, T_1 >, \ldots, < Location_i, T_i >, \ldots, < Location_m, T_m >\}, (1 \le i \le m) \quad (3)$$

where $< Location_i, T_i >$ means that the spatial location of a node in the virtual scene is $Location_i$ at time T_i. This can be regarded as a sampling point on the node's trajectory.

Definition 4 (Interest Cluster)**.** An interest cluster is classified by interest similarity based on the group, trajectory, and other characteristics of a node P_i. Nodes that will request similar scene data are classified into the same cluster, and any node in the cluster can act as the data-supplying source for the others. We set

$$Cluster(T) = \{Cluster \mid sim(P_i, P_j, T) > 0\}, (1 \le i, j \le n) \quad (4)$$

where *Sim( )* is the similarity calculation function.

3.4. Interest similarity

3.4.1. Group Similarity.

The group relationship can be determined by existing logical relationships such as friend or battle relationships, but the group relationship has limited currency because it changes continuously with time t. The formula is as follows:

$$Simg\,(p_i,\ p_j, t) = \begin{cases} 1,\ if\ \ p_i, p_j \in Group(t) \\ 0,\ if\ \ p_i, p_j \notin Group(t) \end{cases} \quad (5)$$

3.4.2. Trajectory Similarity.

The cluster query points in the virtual world are set in the scene-preprocessing stage. Using the similarity of the node location series $TR = \{< l_1, t_1 >, < l_2, t_2 >, \ldots, < l_m, t_m >\}$ and the cluster query point series $Q = \{q_1, q_2, \ldots, q_n\}$, the nodes will be partitioned into diverse clusters. Here, we adopt the *k-nearest neighbor* algorithm to calculate the similarity.

The calculation of trajectory similarities is as follows (Figure 2).

Given the location series TR_i of node $i, TR_i = \{< l_{i1}, t_{i1} >, < l_{i2}, t_{i2} >, \ldots, < l_{im}, t_{im} >\}$ and query points set $Q_j = \{q_{j1}, q_{j2}, \ldots, q_{jn}\}$,

(1) Calculate the mapping distance $\boldsymbol{Mapdist(l_{iu}, q_{jv}, t_{iu})}$ of location points and query points.

Set the spatial distance of location point $\boldsymbol{u}$ and query point $\boldsymbol{v}$ as $dist(l_{iu}, q_{jv}), dist(\ )$; this is the *Euclidean distance*.

For any $l_{iw} \neq l_{iu}$ having $dist(l_{iu}, q_{jv}, t_{iu}) < dist(l_{iw}, q_{jv}, t_{iw}), l_{iu}$ is then called q_{jv}'s mapping on TR_i at the moment t_{iu}; this is denoted as $(q_{jvu}, t_{iu}) \rightarrow TR_i$. Set the mapping distance as

$$Mapdist(l_{iu}, q_{jv}, t_{iu}) = dist((q_{jvu}, t_{iu}) \rightarrow TR_i, q_{jv}, t_{iu}) \quad (6)$$

(2) Calculate the distance $\boldsymbol{SimDist(TR_i, Q_j, t)}$ between node trajectory and query points set.

Given that $X\{x(t_1), x(t_2), \ldots\ldots, x(t_n)\}$ is a monotonically increasing function of time, $X\{Mapdist(l_{i1}, q_{j1}, t_{i1}), \ldots\ldots, Mapdist(l_{in}, q_{jn}, t_{in})\} = X\{dist((q_{jv1}, t_{i1}) \rightarrow TR_i, q_{jv}, t_{i1}), \ldots\ldots, dist((q_{jvn}, t_{in}) \rightarrow TR_i, q_{jv}, t_{in})\}$, where $m > n$, $1 \le v \le n$.

Set the distance of the node trajectory and the query points set as

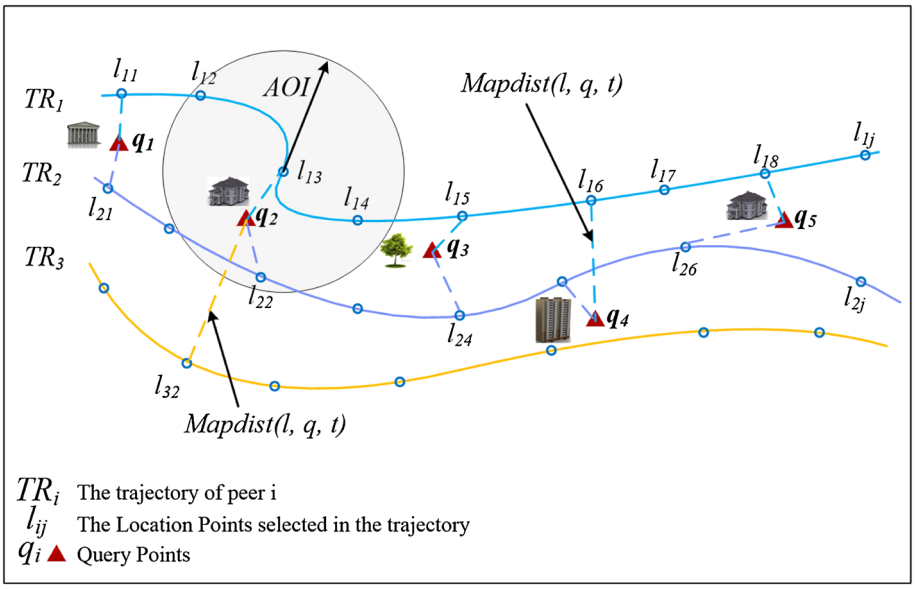


Figure 2. Trajectory similarity calculation.

$$SimDist(TR_i, Q_j, t) = \Sigma Mapdist(l_{in}, q_{jn}, t_{in}) = \Sigma dist((q_{jvn}, t_{in}) \rightarrow TR_i, q_{jv}, t_{in}) \quad (7)$$

where $t = t_{in}$.

(3) Calculate the similarity $\boldsymbol{Simq}(\boldsymbol{p_i}, \boldsymbol{Q_j}, \boldsymbol{t})$ of the node trajectory and query points set.

Given the distance $Dist(TR_i, Q_j, t)$ of trajectory and query points set, set R_{AOI} as the radius of node's AOI.

Set

$$Simq\ (p_i,\ Q_j,\ t) = \begin{cases} 1 - \frac{SimDist(TR_i, Q_j, t)}{n \times R_{AOI}}, & if\ Dist(TR_i, Q_j, t) < R_{AOI} \\ 0 & if\ Dist(TR_i, Q_j, t) > R_{AOI} \end{cases} \quad (8)$$

(4) Calculate the trajectory similarity $\boldsymbol{Simt}(\boldsymbol{p_i}, \boldsymbol{p_j}, \boldsymbol{t})$ between nodes.

Calculate the trajectory similarity between nodes by $\boldsymbol{Simq}(\boldsymbol{p_i}, \boldsymbol{Q_j}, \boldsymbol{t})$, Set

$$Simt(p_i, p_j, t) = Simq(p_i, Q_j, t) \times Simq(p_j, Q_j, t) \quad (9)$$

3.4.3. Interest Similarity.

We can obtain the degree of interest similarity via the degrees of group similarity and trajectory similarity and then construct an interest cluster, as follows.

(1) Calculate the interest similarity $\boldsymbol{Simi}(\boldsymbol{p_i}, \boldsymbol{p_j}, \boldsymbol{t})$ between nodes.

$$Simi(p_i, p_j, t) = \alpha_1 \times Simt(p_i, p_j, t) + \alpha_2 \times Simg(p_i, p_j, t) \quad (10)$$

where $\alpha_1 = S/n\pi R_{AOI}^2$ and $\alpha_2 = 1 - \alpha_1$. (S is the scene area of cluster, n is the number of nodes belonging to the same group, and $R_{AOI}$is the radius of node's AOI.)

(2) Construct the interest cluster $\boldsymbol{Cluster_j(t)}$.

For any $i,j \in [1,n]$, if $Simi(p_i, p_j, t) > 0$, then $p_i, p_j \in Cluster_j(t)$.

4. CONSTRUCTION OF THE NEIGHBOR MESH BASED ON INTEREST CLUSTERS

Based on the interest-similarity algorithm and the cluster classification algorithm proposed in Section 3, we can divide the nodes into several diverse clusters. Because of the varying similarity between nodes, a requester can choose those nodes that have a high similarity degree with itself to construct its neighbor mesh.

Stable Cluster: When the data-supply capacity of a node cluster can sustain the data request demands consistently, we define the cluster as being in a stable condition, and it is known as a stable cluster.

The supply capacity of a node is determined by a synthesis of its network bandwidth, memory capacity, and physical topology. For avatar nodes, however, this involves only the amount of requested data, the amount of response data, and the network latency. In this paper, we proposed the conception of "interest entropy" as the criterion for adjusting a clusters stability.

Degree of Attention: This is the degree of interest by an avatar in the scene model. In a virtual scene, it accords with the scene model's features and the importance of the model for the avatar. Given the degree of attention for each model in the scene, an avatar will first request, download, and render those models with a high degree of attention to enhance the users experience.

Model Unit: This is the minimal independent unit in the scene, as the basic unit for an avatar to request and download. It is set as $u_i, 0 \leq i \leq Unit_{max}$, where $Unit_{max}$ is the maximum number of models in the cluster.

Requested Data: This is the total amount of data requested of all nodes in the cluster over a period Δt, according to the order of the degree of attention. It is set as

$$Request_{\Delta t} = \sum_{i=1}^{n} a_i u_i(d) \tag{11}$$

where u_i represents the model units, a_i represents the number of this models requested, and d represents the degree of attention for this model, $1 \leq i \leq n$.

Response Data: This is the amount of response data for all nodes in the cluster over a period Δt, according to the order of the degree of attention. It is set as

$$Response_{\Delta t} = \sum_{i=1}^{n} b_i u_i(d) \tag{12}$$

where u_i represents the model units, b_i represents the number of this model's response, and d represents the degree of attention for this model, $1 \leq i \leq n$.

Network Latency: This is the difference between the time that the scene model data is received and the time that the node sent the data request. It involves $T_j = T_j(Response) - T_j(Request)$ where $T_j(Response)$ means the timestamp of a request for the jth scene data and $T_j(Request)$ means the timestamp of the response for the jth scene data. If there is no response to the request for the scene data, then set $T_j = \Delta t$. The network latency is normalized as

$$L = \left(1 - \frac{\Sigma T_j}{\Sigma a_i \times T_j^{max}}\right) \tag{13}$$

where $1 \leq j \leq \Sigma a_i$ and T_j^{max} represents the maximum transmission latency.

Definition 5 (Interest Entropy)**.** This is the quantitative indicator used to measure the stability of an interest cluster. It is set as

$$H_{\Delta t} = -lnV_{\Delta t} \tag{14}$$

where $V_{\Delta}t$ are the normalized parameters $V_{\Delta t} = \sum_{i=1}^{n}(b_i/a_i)L$.

The properties of $V_{\Delta t}$ are as follows:

(1) $0 < V_{\Delta t} < 1$,
(2) $V_{\Delta t}$ is continuous and nonnegative,
(3) when $V_{\Delta t}$ increases, $H_{\Delta t}$ decreases, which shows that the responsiveness of the nodes in the cluster better fills the requested demand. If the nodes in the cluster can ensure that the relationship between supply and demand is more balanced, then the nodes will be more orderly, and the cluster will be more stable.

Definition 6 (Entropy Difference)**.** The behaviors and locations of the nodes in the scene are changing constantly with time. Meanwhile, the requested node data is changing, which implies that the stability of a cluster is constantly changing, with a consequent change to the interest entropy. We describe this as

$$\Delta H = H_{\Delta t1} - H_{\Delta t2} = -ln\frac{V_{\Delta t1}}{V_{\Delta t2}} \tag{15}$$

For the period ΔT, we choose n periods Δt from ΔT to calculate the interest entropy and use the variance of the interest entropy to measure the change in the cluster's stability:

$$D(\Delta H) = \frac{\sum_{i=1}^{n}(H_{\Delta ti} - E(H_{\Delta t}))^2}{n} \tag{16}$$

where $E(H_{\Delta t})$ represents the expected value of the interest entropy over Δt.

If, in the period ΔT, $D(\Delta H) < \varepsilon$, this shows that the cluster has been in a stable state during this period. In this case, the nodes in the cluster can achieve the balance between supply and demand in the scene transmission.

Figure 3 illustrates the scene transmission procedure of the nodes in the same cluster; we assign the nodes p_1, p_2, p_3, p_4, p_5 to the same cluster. They enter the same identical region with time sequence for roaming and move in the same main direction. Because of the node's cache is limited, cached data must be constantly updated with the scene data of current AOI. However, the amount of the node's required data is same with each other in the whole roaming process. Our cache updating algorithm will guarantee that the entire region scene can be distributed to the caches of all nodes in this cluster by its capacity. According to the relative time sequence of avatars through the preferred regions, we can divide the nodes into previous nodes and subsequent nodes, and the previous nodes in the cluster will become the potential data-supplying nodes for subsequent nodes. Continuously over time, the later nodes will become the subsequent nodes, and the number of nodes in the cluster will increase constantly. When the variance of interest entropy is $D(\Delta H) < \varepsilon$, we can remove some previous nodes from the cluster to release computing capacity of these modes. In this way, each node in the cluster will maintain a stable and adequate neighbor mesh. Algorithm 1 describes the process of constructing neighbor mesh.

Figure 4 illustrates the construction process for the node neighbor mesh. Here, the cluster is in a stable condition when $\Delta T = 10$ seconds, $\Delta t = 2$ seconds, and $D(\Delta H) < 0.005$. At time T_1, peers p_1, p_2, p_3, p_4 construct a node cluster. Because p_1 is the earliest node to join the cluster, it need not request scene data from the last node p_4. Therefore, p_1 acts only as the parent for p_4, and the other nodes are data-supplying sources for each other. At times T_2 and T_3, peers p_5 and p_6 join the cluster in succession. Because the cluster stability measure is $D(\Delta H) < 0.005$, the cluster has reached a stable status and can remove the first-joining peer p_1 (maybe the nodes have left the region in which other peers gathered). Peers p_5 and p_6 can choose the peers

Algorithm 1. The Construction Algorithm for the Neighbor Mesh

Input: Simg (p_i, S_j, t), Simq (p_i, S_j, t), Simi (p_i, p_m, t)

Output: NeighborMesh$_m$

//Node p_i join in the Cluster$_j$ *//S_j is the super node of Cluster$_j$*

1. ***if*** *Simq* $(p_i, S_j, t) > 0$ *or Simg* $(p_i, S_j, t) = 1$
2. *Insertp_i in Cluster$_j$* *//in descending order of Simq* (p_i, S_j, t)
3. ***for*** *each node in Cluster$_j$*
4. *Insert p_i in NeighborMesh$_m$ of each node* *// in descending order of Simi* (p_i, p_m, t)
5. ***endfor***
6. ***else***
7. *connect p_i with Server* *//If node p_i can not be divided into any cluster, connect it with the server.*
8. ***end if***

//Connect with each other

1. *n=the number of peers needed to connect*
2. sort each *NeighborMesh$_m$* by *Simi (p_i, p_m, t)*
3. ***for*** *the first n nodes of NeighborMesh$_m$*
4. *connect p_i with each node*
5. ***end for***

//Remove node p_i from Cluster$_j$

1. ***while*** *Cluster_threshold*$< \varepsilon$ *//Stability measure of cluster*
2. *select p_i from Cluster$_j$* *//the node of the minimum Simi (p_i, p_m, t)) or the oldest or the minimum memory*
3. *remove p_i from Cluster$_j$*
4. ***for*** *each node from Cluster$_j$*
5. *remove p_i in its NeighborMesh$_m$*
6. ***end for***
7. ***end while***

that have high interest similarity with themselves from the remaining cluster nodes to act as scene data-supplying sources. At times T_6, peers p_7 and p_8 also join the cluster in succession. Similarly, the algorithm can remove peers p_2 from the cluster, which maintains the number of nodes in the stable cluster.

5. PERFORMANCE EVALUATION

Our experiments mainly simulate the virtual scenes and the avatar behaviors in Battle Ground (part of WoW). The experiments were conducted on the simulation platform FloD [12]. FloD is an excellent open-source software project, and we extended it for our experiments as Interest-FloD to include our neighbor node-organizing mechanism.

5.1. Extension of the FloD Platform

FloD mainly considers two moving models, a random model and a cluster model, and adopts that the neighbor discovery mechanism depends on the spatial distance. These two models cannot simulate the avatar behaviors of popular virtual worlds very well. We preprocessed the virtual scene and established a new moving model, *a roaming model*, to simulate the real scenes and avatar behaviors in WoW. We designed the paths in the scene to be similar to the paths in some Battle Ground scenes in WoW and specified the hot and non-hot regions. Nodes can move within the range of different paths at various speeds and can roam randomly from one location to another, and some nodes have a group relationship for some periods (as shown in Figure 5). FloD retains the original Voronoi-based neighbor-discovery mechanism, whereas

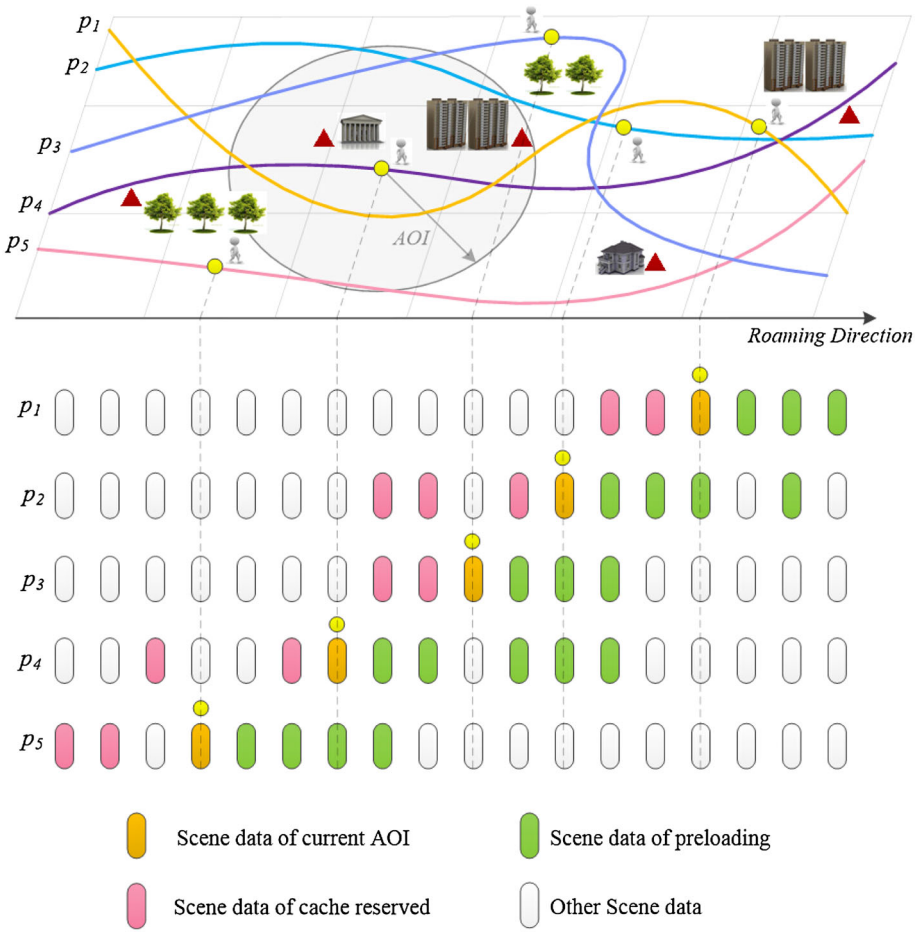


Figure 3. Procedure for scene data loading.

Interest-FloD adopts our neighbor node-organizing mechanism based on avatar interests.

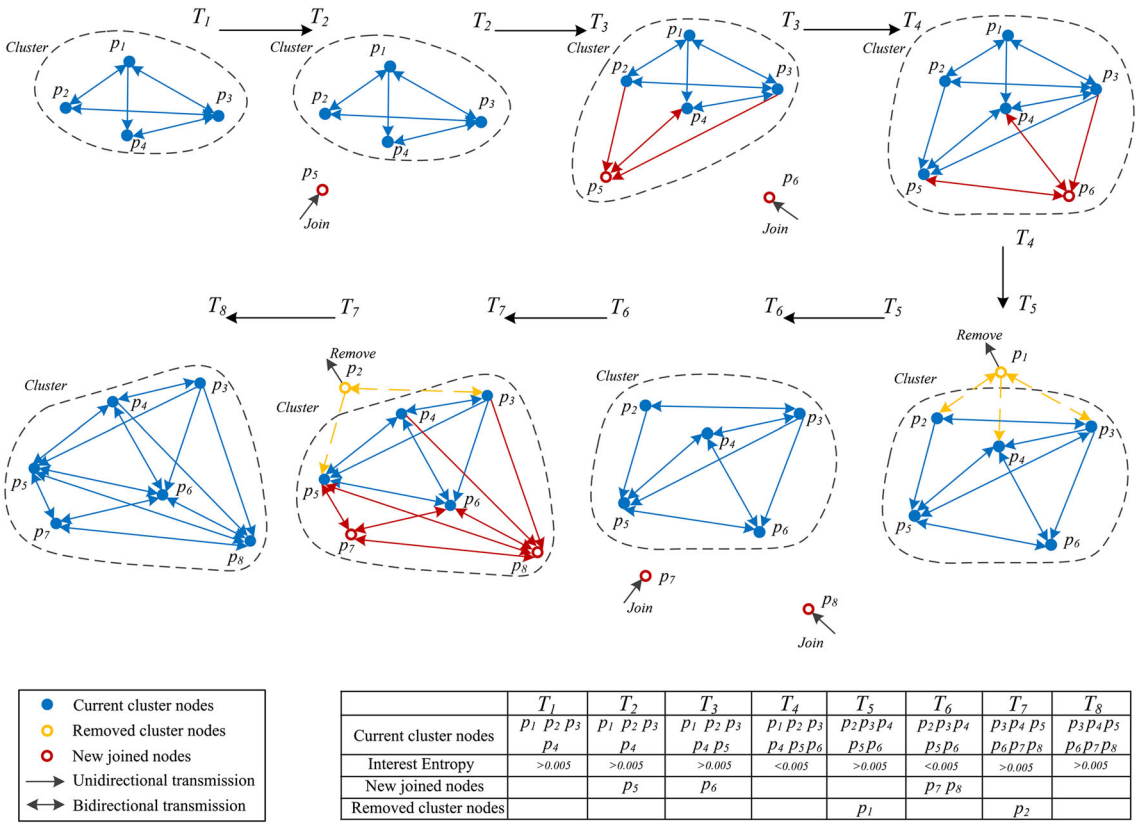
We adopted the performance metrics reported in the literature [12] and conducted a comparative analysis in our experiments. Furthermore, we add two important parameters of scene processing that are the model attention degree and the reusability in the experiments; the experimental parameter values are given in Table I.

Because of the limited timeliness and regional locality with these two characteristics of interest relationships between nodes, experiments with moderate-scale virtual scenes should be able to demonstrate the feasibility of our mechanism fully. Therefore, in each experiment, we varied the number of nodes from 100 to 1000 and set one step as the unit of time.

Furthermore, we build the real DVE based on open-source communication framework (ACE [29]) and rendering engine (OSG [30]); next, we will migrate the experiment to this platform. Figure 6 is a screenshot of our sub-scene with first-person perspective, and the aerial view of this region is shown in Figure 7; the scene characteristics of this sub-scene, such as scene distribution and region attraction, is greatly distinct, and it can adequately represent the current virtual world.

5.2. Average Neighbors

The average neighbors of a node are defined as the average number of neighbor nodes that can provide the requested scene data at different times when this node is roaming in the virtual world. This index is the major guarantee of scene data transmission. According to Figure 8, we can see that the average neighbors for Interest-FloD are clearly higher than that for FloD, and that the number of neighbor nodes is very stable. There are large fluctuations in the average neighbors of FloD. The main reason is that the avatars will sometimes gather in hot regions and spread out in non-hot regions. Because the neighbor-discovery mechanism of FloD mainly discovers the neighbors around a node, the number of neighbor nodes is uncertain across the entire roaming process, which can be considered as a limitation. However, Interest-FloD using the temporal and spatial dimensions to discover neighbors has expanded the search range for neighbor nodes; for example, nodes



	T_1	T_2	T_3	T_4	T_5	T_6	T_7	T_8
Current cluster nodes	$p_1\ p_2\ p_3\ p_4$	$p_1\ p_2\ p_3\ p_4$	$p_1\ p_2\ p_3\ p_4\ p_5$	$p_1\ p_2\ p_3\ p_4\ p_5\ p_6$	$p_2\ p_3\ p_4\ p_5\ p_6$	$p_2\ p_3\ p_4\ p_5\ p_6$	$p_3\ p_4\ p_5\ p_6\ p_7\ p_8$	$p_3\ p_4\ p_5\ p_6\ p_7\ p_8$
Interest Entropy	>0.005	>0.005	>0.005	<0.005	>0.005	<0.005	<0.005	>0.005
New joined nodes		p_5	p_6			$p_7\ p_8$		
Removed cluster nodes					p_1		p_2	

Figure 4. Construction of neighbor mesh.

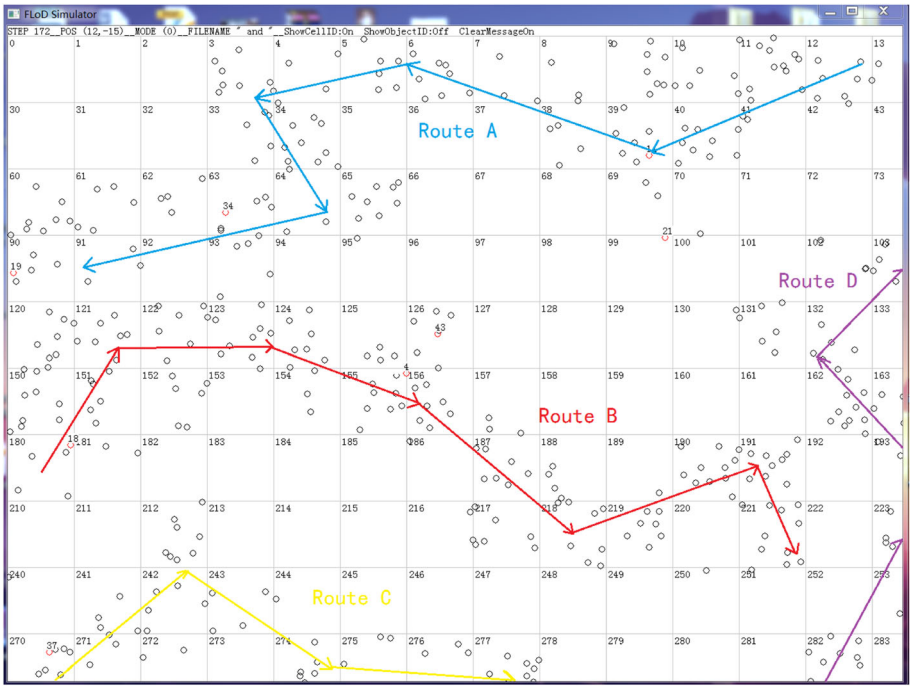


Figure 5. Interest-FloD simulator.

Table I. Experimental parameters.

Parameter	Value
World dimension (units)	3000 × 3000
Cell size (units)	100 × 100
Node num	100–1000
Time steps	1000
AOI radius (units)	75
Base piece per cent (%)	20
Incremental piece size (byte)	1600
Model attention degree	0–1
Model reusability	1–10

AOI, area of interest.

that have traversed a region previously may become the neighbors of subsequent nodes.

5.3. Fill Ratio

The fill ratio is defined as the ratio of the visible scene data that has currently been received to the scene data that was requested. According to Figure 9, we can see that Interest-FloD is superior to FloD with respect to the fill ratio index. The main reason is that a fluctuating number of neighbors leads to an unstable number of scene data-supplying nodes, which causes fluctuations in the fill ratio at different times and lowers the average fill ratio. For Interest-FloD, stable neighbor numbers guarantee adequate data-supplying resources to give a higher fill ratio. However, we can see that the average fill ratio for Interest-FloD is below 90%, mainly because we restricted the node number of clusters. When the node number exceeds the threshold, regardless of the cluster in the stability status, we will divide the cluster into two new clusters in accordance with the order of the nodes joining the cluster, for reducing the amount of message exchange within the cluster nodes.

5.4. Server Request Ratio

The server request ratio is defined as the ratio of the scene data that a node requests from servers divided by the total received data in the entire transmission process. As

Figure 6. First-person perspective of our scenes.

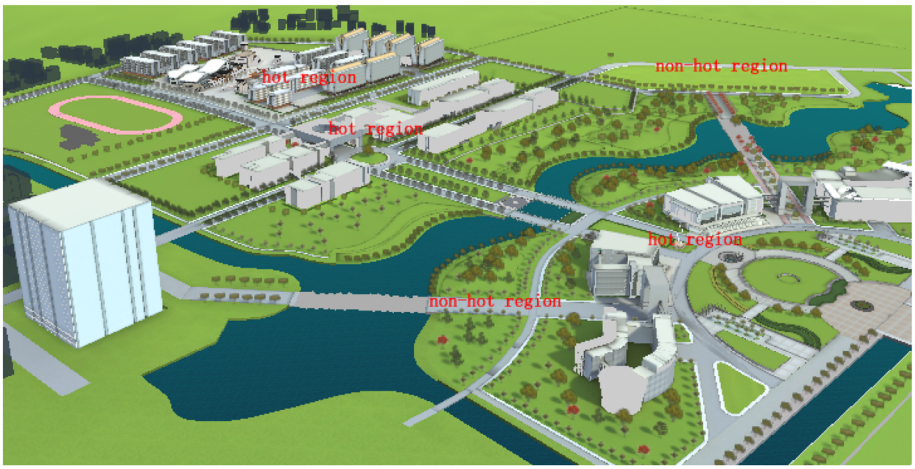


Figure 7. Scene distribution of one sub-scene.

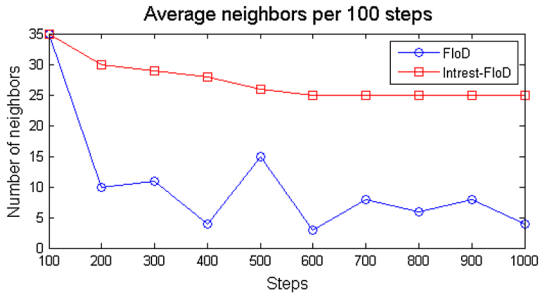


Figure 8. The comparison of the average number of neighbor nodes.

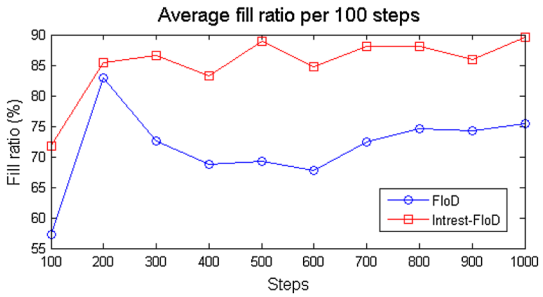


Figure 9. The comparison of fill ratios for FloD and Interest-FloD.

Figure 10. The comparison of server request ratios for FloD and Interest-FloD.

shown in Figure 10, the average server request ratio for Interest-FloD is about 5%, which is lower than the 20% for FloD. This situation occurs mainly because the stable neighbor mesh guarantees responses to data requests. At the same time, both Interest-FloD and FloD need to request a certain amount of data from servers. This is mainly because the amount of request data is affected by many factors, including the distribution of the scene data, the movement speed of the node, and the variable upload bandwidth of neighbor nodes. Therefore, it is reasonable to have to request a certain amount of scene data from a server.

5.5. Base Latency

The scene data comprises a base mesh and a series of incremental meshes after the stream processing based on progressive mesh construction. The base mesh is the first to be transmitted in the transmission process. Base latency is defined as the duration between requesting the base mesh data and receiving them. As shown in Figure 11, Interest-FloD outperforms FloD in this respect. The main reason is that the quantity of neighbors in FloD is limited, with most of them gathered around the requester. The requester needs to query the data from the neighbors continuously, and the performance of these neighbors, in terms of storage capability and network bandwidth, may exhibit large differences, which may increase the network delay. For Interest-FloD, however, the relatively stable and optimal neighbor mesh maintained by each node will enable the neighbors with better performance to supply data for the requester. The network delay will therefore be lower.

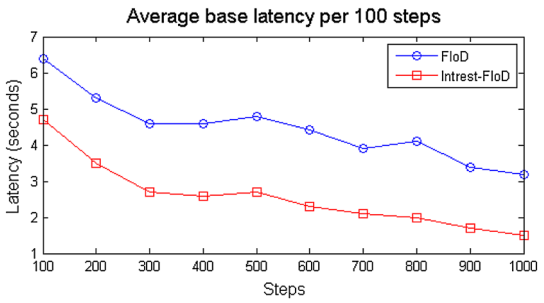


Figure 11. The comparison of base latency for FloD and Interest-FloD.

6. CONCLUSIONS

To address the churn problem in neighbor tables during the process of transmitting large-scale virtual scenes in P2P networks, this paper has proposed a neighbor-organizing mechanism based on avatar interest. Our main contributions are as follows: (i) we have applied avatar behavior analysis to construct neighbor-organizing mechanism for large-scale scene transmission, (ii) we have extracted and quantified the characteristics of avatar common behaviors in popular virtual worlds and proposed an interest-similarity algorithm for avatar behaviors, and (iii) we have proposed the concept of interest entropy to measure the disorder degree of cluster and presented an algorithm for constructing a stable neighbor mesh. Our experimental results have shown that the proposed neighbor-organizing mechanism can mitigate the churn problem in neighbor tables efficiently, reduce the message exchange between nodes, and alleviate the network load.

Our future work will focus on (i) extracting multidimensional characteristics of avatar behaviors and improving the accuracy of the similarity computation, (ii) proposing an efficient scene management mechanism combines with avatar behaviors analysis, and (iii) proposing a scene data-dispatch mechanism and cache-update mechanism based on interest clusters.

ACKNOWLEDGEMENTS

The work was supported by the National Science Foundation of China (No. 61272270), National Twelfth Five-year Plan Major Science and Technology Project (No. 2012BAC11B00-04-03), Special Research Fund of Higher College's Doctorate (No. 20130072110035), Key Science and Technology Project of Jilin (No. 20140204088GX), and Changbai Valley Talent Plan of Changchun National Hi-Tech Industrial Development Zone (No. 3-2013006).

REFERENCES

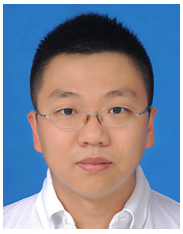
1. Zhou S, Yoo I, Benes B, Chen G. A hybrid level-of-detail representation for large-scale urban scenes rendering. *Computer Animation and Virtual Worlds* 2014; **25**(3-4): 243–253.
2. Tencent. Arche age, 2015. http://age.qq.com. [Accessed on 8 February 2015].
3. Foundation Wikimedia. Second life, 2015. http://en.wikipedia.org/wiki/Second_Life. [Accessed on 15 January 2015].
4. Yahyavi A, Kemme B. Peer-to-peer architectures for massively multiplayer online games: a survey. *ACM Computer Surveys* 2013; **46**(1): 9:1–9:51.
5. Buyukkaya E, Abdallah M, Simon G. A survey of peer-to-peer overlay approaches for networked virtual environments. *Peer-to-Peer Networking and Applications* 2015; **8**(2): 276–300.
6. Legtchenko S, Monnet S, Sens P, Muller G. Relaxdht: a churn-resilient replication strategy for peer-to-peer distributed hash-tables. *ACM Transactions on Autonomous and Adaptive Systems* 2012; **7**(2): 28:1–28:18.
7. Li J, Stribling J, Gil TM, Morris R, Kaashoek MF. Comparing the performance of distributed hash tables under churn. In *Peer-to-Peer Systems III*. Springer: La Jolla, CA, USA, 2005; 87–99.
8. Knutsson B, Lu H, Xu W, Hopkins B. Peer-to-peer support for massively multiplayer games. In *Infocom 2004. Twenty-Third Annual Joint Conference of the IEEE Computer and Communications Societies*, Vol. 1, IEEE: Hong Kong, China, 2004.
9. Bharambe AR, Agrawal M, Seshan S. Mercury: Supporting scalable multi-attribute range queries. In *Proceedings of the 2004 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*. SIGCOMM '04. ACM: Portland, Oregon, USA, 2004; 353–366.
10. Cordasco G, De Chiara R, Erra U, Scarano V. Some considerations on the design of a p2p infrastructure for massive simulations. In *ICUMT'09 International Conference on Ultra Modern Telecommunications & Workshops, 2009*. IEEE: St. Petersburg, Russia, 2009; 1–7.
11. Schmieg A, Stieler M, Jeckel S, Kabus P, Kemme B, Buchmann A. psense-maintaining a dynamic localized peer-to-peer structure for position based multicast in games. In *P2P'08. Eighth International Conference on Peer-to-Peer Computing, 2008*. IEEE: Aachen, Germany, 2008; 247–256.
12. Hu SY, Huang TH, Chang SC, Sung WL, Jiang JR, Chen BY. Flod: A framework for peer-to-peer 3d streaming. In *Infocom 2008. the 27th Conference on Computer Communications. IEEE*. IEEE: Phoenix, AZ, USA, 2008.
13. Shen S, Brouwers N, Iosup A, Epema D. Characterization of human mobility in networked virtual environments. In *Proceedings of Network and Operating System Support on Digital Audio and Video Workshop*. ACM: Singapore, 2014; 13.
14. Park SI, Quek F, Cao Y. Simulating and animating social dynamics: embedding small pedestrian groups in crowds. *Computer Animation and Virtual Worlds* 2013; **24**(3-4): 155–164.
15. La CA, Michiardi P. Characterizing user mobility in second life. In *Proceedings of the First Workshop on Online Social Networks*. WOSN '08. ACM: Seattle, WA, USA, 2008; 79–84.
16. Suznjevic M, Matijasevic M. Player behavior and traffic characterization for MMORPGS: a survey. *Multimedia Systems* 2013; **19**(3): 199–220.
17. Pittman D, GauthierDickey C. A measurement study of virtual populations in massively multiplayer online games. In *Proceedings of the 6th ACM SIGCOMM Workshop on Network and System Support for Games*. ACM: Melbourne, Australia, 2007; 25–30.
18. Carlini E, Coppola M, Ricci L. Evaluating compass routing based AOI-cast by MOGS mobility models. In *Proceedings of the 4th International ICST Conference on Simulation Tools and Techniques*. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering): Barcelona, Spain, 2011; 328–335.
19. Legtchenko S, Monnet S, Thomas G. Blue banana: resilience to avatar mobility in distributed MMOGS. In *2010 IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*. IEEE: Chicago, IL,USA, 2010; 171–180.
20. Miller JL, Crowcroft J. Avatar movement in world of warcraft battlegrounds. In *Proceedings of the 8th Annual Workshop on Network and Systems Support for Games*. IEEE Press: Paris, France, 2009; 1.
21. Varvello M, Ferrari S, Biersack E, Diot C. Exploring second life. *IEEE/ACM Transactions on Networking (TON)* 2011; **19**(1): 80–91.

22. Liang H, De Silva RN, Ooi WT, Motani M. Avatar mobility in user-created networked virtual worlds: measurements, analysis, and implications. *Multimedia Tools and Applications* 2009; **45**(1-3): 163–190.
23. Yu AP, Vuong ST. 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*. ACM: Stevenson, Washington, USA, 2005; 99–104.
24. Buyukkaya E, Abdallah M, Cavagna R. Vorogame: a hybrid p2p architecture for massively multiplayer games. In *CCNC 2009. 6th IEEE Consumer Communications and Networking Conference, 2009*. IEEE: Las Vegas, NV, USA, 2009; 1–5.
25. Wang G, Wang K. An efficient hybrid p2p mmog cloud architecture for dynamic load management. In *2012 International Conference on Information Networking (ICOIN)*. IEEE: Bali, Indonesia, 2012; 199–204.
26. Varvello M, Biersack E, Diot C. Dynamic clustering in Delaunay-based p2p networked virtual environments. In *Proceedings of the 6th ACM SIGCOMM Workshop on Network and System Support for Games*. ACM: Melbourne, Australia, 2007; 105–110.
27. Sripanidkulchai K, Maggs B, Zhang H. Efficient content location using interest-based locality in peer-to-peer systems. In *Infocom 2003. Twenty-Second Annual Joint Conference of the IEEE Computer and Communications. IEEE Societies*, Vol. 3, IEEE: San Francisco, CA, USA, 2003; 2166–2176.
28. Xiao L, Liu Y, Ni LM. Improving unstructured peer-to-peer systems by adaptive connection establishment. *IEEE Transactions on Computers* 2005; **54**(9): 1091–1103.
29. The ADAPTIVE Communication Environment. Ace 2015, 2015. http://www.cs.wustl.edu/~schmidt/ACE.html. [Accessed on 8 May 2014].
30. OpenSceneGraph. Osg 2015, 2015. http://www.openscenegraph.org. [Accessed on 10 June 2014].

AUTHORS' BIOGRAPHIES

Mingfei Wang is a Ph.D. student in the School of Software Engineering at Tongji University. He received his M.S. degree from China University of Mining and Technology and B.S. degree from Zhengzhou University. His current research interests include distributed virtual environment and data mining.

Jinyuan Jia is a Professor in the School of Software Engineering and Director of Graphics and Image Research Center at Tongji University. He received his Ph.D. degree in Computer Science from Hong Kong University of Science and Technology and M.S. Degree from Jilin University. His research is primarily in the areas of computer graphics, geometric modeling, Web 3D, and game engine.



Ning Xie is an Assistant Professor in the School of Software Engineering at Tongji University. He received his Ph.D. degree and M.S. degree from the Department of Computer Science at Tokyo Institute of Technology in 2009 and 2012, respectively. In 2012, he was appointed as a Research Associate in the same institute. His is recently concentrating on the theory and application of machine learning, computer graphics, and computer vision.

Chenxi Zhang is a Professor in the School of Software Engineering at Tongji University. He received his Ph.D. degree and B.S. degree from the National University of Defense Technology. His research interests include computer architecture and distributed computing.