VoroGame : A Hybrid P2P Architecture for Massively Multiplayer Games

Eliya Buyukkaya, Maha Abdallah, Romain Cavagna
Laboratoire d'Informatique de Paris 6
Université Paris 6
Paris, France
{Eliya.Buyukkaya, Maha.Abdallah, Romain.Cavagna}@lip6.fr

***Abstract*—Peer-to-peer (P2P) architectures have recently become very popular in massively multiplayer games (MMGs). While P2P gaming offers high scalability compared to client/server architectures, it introduces several major issues related to data distribution and game state consistency. In this paper, we report our initial version of VoroGame, a P2P architecture for MMGs that addresses these issues by combining a structured P2P overlay based on a distributed hash table (DHT) for data distribution, with a Voronoi diagram used for virtual game world decomposition and semantic overlay support. The resulting hybrid architecture enables a fully distributed management of data and state information, and ensures efficient dissemination of game state updates to relevant peers.**

***Keywords*—Massively multiplayer games, peer-to-peer systems, distributed hash tables, voronoi diagram, interest management, data distribution, state management.**

I. INTRODUCTION

Massively multiplayer games (MMGs) are becoming extremely popular nowadays. MMGs are traditionally supported by a client/server architecture where every player communicates with a server that stores the whole game state and is responsible of broadcasting state changes to all interested players. However, centralized architectures lack flexibility, and can lead to high communication and computation overhead on the server, which quickly becomes a bottleneck during peak loads. To overcome these problems inherent to centralized solutions, P2P overlay networks are emerging as a promising architecture for MMGs [2-8, 11]. However, exploiting P2P schemes in MMGs is not straightforward, and several challenging issues related to data distribution and game state consistency should be considered.

In a typical game, a player sees a portion of the game world where it can perform actions (i.e., moving, manipulating objects, communicating with other players, etc.). This portion of the game world, namely the perception the player has of the world, called Area of Interest (AOI), is often based on proximity, and is modeled as a sphere around the player. A player is only interested in the activities happening within its AOI. In this context, the key issue in P2P gaming is to dynamically partition the virtual game world and distribute game data evenly among all the participating peers (players), and to maintain game state consistency by sending to each peer only relevant update messages (i.e., update information related to other peers or objects residing inside its AOI).

In this paper, we propose VoroGame, a hybrid P2P architecture for MMG applications. Our game architecture combines a structured P2P overlay based on a distributed hash table (DHT) (e.g., [9, 10]) with an unstructured P2P overlay based on a Voronoi diagram [1]. More precisely, a DHT is used for data distribution and storage among all peers, while a Voronoi diagram is used for virtual game world decomposition and semantic overlay support. The intuition behind this choice is based on the fact that in MMGs, players and objects evolve very quickly in the game world. While it is natural that the virtual world decomposition among the peers, which translates their dynamically changing AOI, captures this evolution, it is not feasible from a performance perspective to couple the storage and management of data objects with the continuously evolving nature of the game world.

In the light of the above, our hybrid architecture exploits a DHT to distribute game data evenly and randomly among all participating peers, independently from their positions in the virtual world. A Voronoi diagram is then used to efficiently disseminate game state updates only to relevant peers by making effective use of the semantic relations between peers and objects (based on their virtual world positions).

The rest of the paper is organized as follows. Section 2 defines and discusses the VoroGame system model. Section 3 details several issues related to the VoroGame architecture. Section 4 presents related works. Finally, section 5 concludes this work and points out some of our future perspectives.

II. VOROGAME: A HYBRID SYSTEM ARCHITECTURE

In VoroGame, a peer is assigned two tasks. The first task, related to the structured overlay, is to store data objects mapped to it by the DHT. The second task, related to the unstructured overlay, is to be responsible for a specific zone

978-1-4244-2309-5/09/$25.00 ©2009 IEEE

of the game world in which the peer is located, and to determine the set of other peers concerned by its own state change (i.e., position change), or by updates related to objects residing in its zone (i.e., the set of peers whose AOI cover these changes). Therefore, there are two responsible peers for an object in the game. More precisely, the dissemination of an object's updates is done with the help of two responsible peers of the object, one in the virtual world (i.e., on the Voronoi diagram) and the other on the DHT. The Voronoi responsible peer of an object determines the set of peers whose AOI contains the object and sends this set to the DHT responsible peer. The DHT responsible peer then disseminates the object's updates to the set of peers determined by the Voronoi responsible.

A. Structured overlay

Peers and objects are randomly mapped to a DHT address space. In this work, we assume no particular DHT topology structure. However, we assume that the address space is dynamically partitioned among all the peers in the system such that every peer manages a zone in the address space. An object is then assigned to (stored at) the peer managing the zone to which the object is mapped. This peer is called DHT responsible of the object. The mapping of peers and objects to the DHT space is not affected by changes occurring in the virtual world (i.e., position changes of objects and peers). The DHT responsible of an object remains the same even if in the game world either the object has moved or the nearest peer to the object has changed. Therefore, there is no unnecessary network utilization to transfer objects between peers due to a high dynamism in the virtual world.

The objects managed by a peer may reside in different regions of the game world. Thus, a peer manages game objects residing neither necessarily within its area of interest nor within a particular area of the game world. In our architecture, the structured overlay is used for object distribution and storage. The peers concerned by a state update of an object are automatically notified by the system. Therefore, no DHT object lookup is performed.

B. Unstructured overlay

We partition the game world into regions based on the Voronoi algorithm [1]. The peer whose position coincides with the center point of a region is responsible of that region. In the game world, the nearest peer to a game object is called the object's Voronoi responsible that possesses a copy[1] of the object. A peer, say p, maintains two data structures: (1) an object list that contains the objects residing inside p's responsibility region (i.e., objects for which p is Voronoi responsible), and (2) a peer list that contains the peers that need to be informed about the game state changes occurring in p's responsibility region (i.e., peers to which these changes are visible). Upon a change in peer p's state, p sends the necessary update information about its state to the peers in its peer list whose AOI contains its position in the virtual world (i.e. peers to which p is visible).

In addition, for each object in its object list, p sends to the object's DHT responsible (an update of) the list of peers to which the object is visible. Thus, the object's DHT responsible disseminates, whenever needed, the necessary update information about the object's state to the peers in the list sent by the object's Voronoi responsible. In the face of peer p's movement, p informs the related peers in its peer list about its position change. The boundaries of the responsibility region are changed; however, the positions of the objects in the world remain the same. Therefore, p's object list might change due to its movement. If so, p gives up the responsibility of objects now lying in the region of another peer. In addition, p takes over the responsibility of objects now part of its new region and informs the DHT responsible of the newly joined objects about Voronoi responsibility change. Then, p reconstitutes its peer list. The subsequent update process about p and any object in the object list is achieved based on the updated peer list.

Region visibility. Following the same approach we proposed in [2], we identify all peers concerned with the updates that have occurred in a region (i.e., the area where the changes inside the region are visible) in the following way. For a region with a responsible peer p, we first define a minimum *convex polygon* P(p) that contains all objects within p's AOI as well as p (see Figure 1). Only the objects inside p's AOI are taken into account because the objects outside this region cannot be visible to any other peer. Based on this polygon P(p), we then define *polygon visibility region* VP(p) which indicates the visibility area of P(p) (i.e., the visibility area of the items inside p's region). If the radius of p's AOI[2] is r, the definition of the polygon visibility region VP(p) can be given as follows:

$$VP(p) = \left\{ x \in R^2 \,\middle|\, dist(x,i) \le r \quad \forall i \in P(p) \right\}$$

In Figure 1, for a given peer p, the polygon P(p) and the polygon's visibility region VP(p) are respectively represented by the dark-gray area and the convex curve surrounding P(p). The peers within VP(p) (whose zones are colored light-gray) are potential peers to be informed about the updates caused by the state changes of peer p or the objects in P(p), meaning that the state updates occurring inside the region are visible either partly or fully to these peers within VP(p) that constitute p's peer list.

[1] A copy of an object at a peer contains only the object's state information that concerns the peer.

[2] We assume that peers have the same AOI size, which is also equal to the size of an object's visibility area.

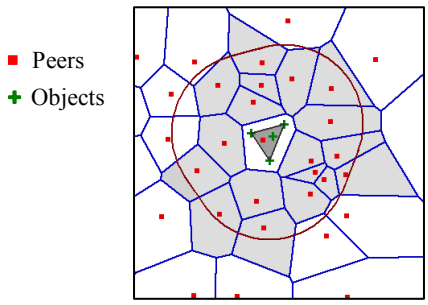

Figure 1. Polygon (dark-gray area) and Polygon Visibility Region (convex curve surrounding the Polygon).

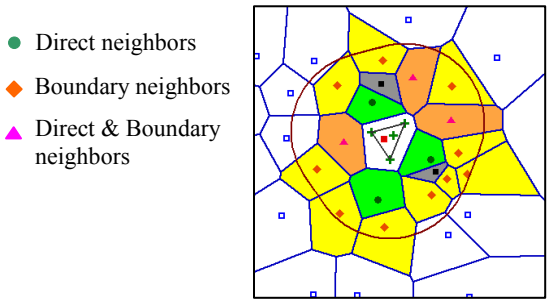

Figure 2. Neighborhood relations.

Neighborhood relations. Similarly to [2], there are two different types of neighborhood relations between peers in the unstructured overlay. For a given peer p, a peer q is called direct neighbor if there is a common edge between the regions of p and q. Besides, a peer r is called boundary neighbor if (1) r lies inside polygon's visibility region VP(p), and (2) at least one direct neighbor of r lies outside VP(p). Note that a peer s can be both direct and boundary neighbor of p if (1) s's region has a common edge with p's region, and (2) s has at least one direct neighbor lying outside VP(p) (see Figure 2).

III. GAME MANAGEMENT IN VOROGAME

A. Data distribution

Immutable game data (i.e., landscape information) are installed as part of the game software on each peer. Mutable data are evenly distributed among all peers by the DHT. The data of a game object o are possessed by o's DHT responsible peer. In the virtual world, o's Voronoi responsible peer as well as the peers whose AOI contains o's position (i.e., to whom o is visible) possess a copy of o. Thus, o has two responsible peers, (1) the DHT responsible and (2) the Voronoi responsible, which know each other. In the face of responsible peer change, o's new responsible peer informs the other responsible about this responsibility change. Using a DHT overlay provides a random and balanced distribution of data among all participating peers independently from their positions in the virtual world, and avoids unnecessary object data transfer between peers due to a high dynamism of the virtual world. Our system allows also peers to insert objects into the game virtual world. A peer p that inserts an object o will be the first Voronoi responsible of o. Peer p then finds the peer who will be o's DHT responsible by mapping object o to the DHT structure.

B. State management

The state management of a peer is achieved by the peer itself by sending the necessary update information about its state change to the peers in its peer list whose AOI contains its position in the virtual world. The state management of an object o is done cooperatively by o's DHT and Voronoi responsible peers. Whenever movements involving o occur (cf. section 3.3), o's Voronoi responsible sends to o's DHT responsible (an update of) the list of peers to which o is visible. o's DHT responsible then disseminates through the DHT routing infrastructure the necessary update information about o's state to the peers in the list determined by o's Voronoi responsible.

C. Movement

Peer movement. When a peer p moves, p informs the related peers in its peer list about its position change. Towards this end, p first informs its direct neighbors in the semantic overlay, i.e., in the game world. As a consequence of p's position change, direct neighbors have to recalculate the boundaries of their regions. Some direct neighbors may not be directly connected to p anymore, while some other peers who have not been directly connected may end up as direct neighbors. p also informs all the peers whose AOI contains p's position (i.e., peers to whom p is visible).

Peer p recalculates the boundaries of its Voronoi region. Since the boundaries of the region are changed, some objects may now lie in the region of any direct neighbor. If there is any object which is not lying in p's region anymore (i.e., the object is no longer managed by p due to position change), then p hands on the responsibility of the object to its direct neighbor. From that point on, the peer into which region the object has newly entered adds the object to its object list and becomes the Voronoi responsible of the object. Moreover, p is informed, if needed, by its direct neighbors about any newly entered objects into its region. Consequently, p adds these objects to its object list and takes over their Voronoi responsibility. Each time a peer takes over the responsibility of new objects informs the objects' DHT responsibles about the Voronoi responsibility change.

Also, p examines if the polygon P(p) has changed. If so, p recalculates P(p) as well as the polygon visibility region VP(p). Moreover, p updates its peer list. Towards this end, p removes from its peer list the peers that are no longer in VP(p), if they are not direct neighbors. Furthermore, to detect newly entered peers into VP(p), p asks its boundary neighbors to check if any of their direct neighbors has entered into VP(p) or has become p's direct neighbor. If so, p adds these peers

into the peer list, and reconstitutes the peer list and neighborhood relations.

Object movement. In the face of an object *o*'s movement, the Voronoi responsible peer *p* sends to *o*'s DHT responsible the updated list of peers to which *o* is visible, whereby *o*'s DHT responsible disseminates *o*'s state update to all the peers in the list. If *o*'s movement induces a change of the polygon P(*p*), then in order to detect peers to whose AOI *o* has newly entered, *p* asks its boundary neighbors to check if any of their direct neighbors has entered into the updated VP(*p*). If *o*'s movement induces a change of the region in which *o* lies, then *o*'s Voronoi responsible becomes the peer owning the region in which the object now lies. *o*'s old and new Voronoi responsible peers reconstruct their Polygons and Polygon Visibility Regions, and update their peer lists. Object *o*'s DHT responsible is also informed about the Voronoi responsibility change.

IV. Related works

Although multiplayer P2P networked gaming is becoming an important and popular application, works that consider data management issue in this context are still at their beginning.

In [8], the authors propose a distributed solution for video gaming based on a distributed client/server architecture (or equivalently, a super-peer architecture). In this solution, the game world is partitioned into fixed-size regions with all peers lying in a particular region forming a single group that is managed by a unique server node. A server node is then responsible of collecting state changes occurring in its region and disseminating these changes to all the members of its group. Server nodes are randomly chosen and connected to each others based on a distributed hash table. This solution clearly suffers from classical performance, scalability, and robustness problems intrinsic to client/server architectures. Indeed, in dynamic game settings where all objects and nodes might end up in a single region managed by a single server, this solution does not represent a good scalable candidate for fully distributed peer-to-peer gaming.

In [11], the authors propose a similar super-peer architecture with, however, the game world being partitioned into hexagons and super peers being connected following an unstructured overlay network. Similarly to [8], this solution also suffers from the same problems related to client/server architectures.

The work initially presented in [4] and later extended in [5] discusses a Voronoi-based partitioning of space in the context of networked virtual environments. [4] does not deal with data management issues and therefore, is not an appropriate solution for game applications where mutable and immutable objects' data are an intrinsic part of the game world. In [5], the authors deal with the state management issue; however, they use for objects updates dissemination the same state management algorithm used for peers updates dissemination proposed in [4]. This unfortunately leads to situations where some peers cannot be informed about state updates of objects they are interested in. Indeed, since a peer *p* knows only about other peers inside its AOI, if any of *p*'s objects is seen by peers that are not inside its AOI, then *p* cannot disseminate the object's state update to those interested peers.

In [2], we proposed a data management algorithm in the context of Voronoi-based decomposition of the game world. However, since data distribution is tightly coupled with the game world decomposition, a peer responsible of an object in the virtual world also possesses the object's data. This leads to important communication overhead where the object is continuously transferred and reassigned between peers due to the high dynamism of the virtual world and the speed at which the players evolve. Consequently, the discussed solution does not provide any protection against cheating. The present work builds on our previous work, and proposes a P2P gaming architecture designed with all these issues in mind.

V. Conclusion

In this paper, we presented the initial design of VoroGame, a hybrid P2P architecture for massively multiplayer games. Our solution exploits a distributed hash table to distribute game data randomly and evenly among all the participating peers. Locality of interest and semantic proximity between peers and objects are then exploited to build a semantic overlay based on a Voronoi diagram, by which game state updates are efficiently disseminated only to relevant peers. To the best of our knowledge, our solution is the first to offer a fully distributed P2P state and data management architecture, while exploiting the locality of interest of the players for efficient state updates dissemination support.

VoroGame is currently being implemented as open source for evaluation and testing in real game applications.

As a future work, it would be interesting to consider different parameters, such as different object visibility levels depending on object's size, and peers' energy levels so as to maximize the network life.

Acknowledgment

This work is supported by the MAD Games project of the ANR RIAM program of the French Ministry of Research.

REFERENCES

[1] F. Aurenhammer. Voronoi diagrams—a survey of a fundamental geometric data structure. *ACM Computing Surveys*, 23(3):345-405, 1991.

[2] Buyukkaya, E. and Abdallah, M. Data Management in Voronoi-based P2P Gaming. In Proceedings of IEEE International Workshop on Digital Entertainment, Networked Virtual Environments, and Creative Technology (Las Vegas, NV, USA, January 10-12, 2008). CCNC'08. 1050-1053.

[3] D. Frey, J. Royan, R. Piegay, A.-M. Kermarrec, E. Anceaume, and F. Le Fessant. Solipsis: A Decentralized Architecture for Virtual Environments. In *Proc. of the International Workshop on Massively Multiuser Virtual Environments* (MMVE), 29-33, March 2008.

[4] S. Y. Hu, J. F. Chen, and T. H. Chen. VON: A Scalable Peer-to-Peer Network for Virtual Environments. In *IEEE Network*, 20(4), 22-31, July/August 2006.

[5] S. Y. Hu, S. C. Chang, and J. R. Jiang. Voronoi State Management for Peer-to-Peer Massively Multiplayer Online Games. In *Proc. of the IEEE International Workshop on Networking Issues in Multimedia Entertainment*, 1134-1138, January 2008.

[6] 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. of the ACM SIGCOMM Workshop on Network and System Support For Games*, 116-120, August 2004.

[7] J. Keller and G. Simon. Solipsis: A massively multi-participant virtual world. In *Proc. of the International Conference on Parallel and Distributed Techniques and Applications*, 262-268, June 2003.

[8] B. Knutsson, H. Lu, W. Xu, and B. Hopkins. Peer-to-peer support for massively multiplayer games. In *Proc. of IEEE INFOCOM,* 96-107, March 2004.

[9] A. Rowstron and P. Druschel. Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer system. *In Proc. of the IFIP/ACM International Conference on Distributed Systems Platforms (Middleware)*, 329-350, November 2001.

[10] I. Stoica, R. Morris, D. Liben-Nowell, D.R .Karger, M.F. Kaashoek, F. Dabek, and H. Balakrishnan. Chord: A scalable peer-to-peer lookup protocol for internet applications. *IEEE/ACM Transactions on Networking*, 11(1):17-32, 2003.

[11] A. Yu and S. T. Vuong. MOPAR: a mobile peer-to-peer overlay architecture for interest management of massively multiplayer online games. In *Proc. of the ACM international Workshop on Network and Operating Systems Support For Digital Audio and Video*, 99-104, June 2005.