Data Management in Voronoi-based P2P Gaming

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

Abstract—**This paper proposes a fully distributed architecture to support massively multiplayer games (MMGs) on a peer-to-peer (P2P) overlay network. P2P networks provide a scalable and robust solution for MMGs. Furthermore, similarly to real world interactions, MMG players interact with their surrounding, i.e., with visible players and objects in the virtual world. Thus, self-organization of peers based on their visibility property is essential for efficient overlay network support in MMGs. Also, the overlay network must be re-organized consistently in the face of players' position changes. In this paper, we achieve these goals by designing a peer-to-peer overlay network based on the Voronoi diagram. Each player has a responsibility region that includes nearby objects, and is responsible for updates' dissemination occurring in this region to provide game state consistency among the players who are concerned by the updates. As a proof of concept, we simulated a simple game application to demonstrate the feasibility of our architecture.**

Keywords-Massively multiplayer games; peer-to-peer systems; Voronoi diagram; interest management; data management

I. INTRODUCTION

Massively multiplayer games (MMGs) are becoming popular nowadays. In MMGs, a huge number of players are in the same virtual world where they play their roles, change positions and interact with each others and with other objects in the world. MMGs are traditionally supported by a client-server architecture, where every player communicates with a central server. The server is then responsible of broadcasting game state changes to all players in the same game zone. However, a server can serve a limited number of players because of its resource constraints on communication as well as computation. As a result, the overall scalability is strictly limited by the server capacity. Moreover, this approach is subject to a single point of failure and becomes a bottleneck during high network utilization. To overcome the limited scalability and robustness problems inherent to centralized solutions, P2P overlay networks are emerging as a promising architecture for MMGs [2, 3, 4, 5, 6, 7, 8]. In general, a P2P overlay network is self-organized as well as self-repaired without any centralized control, and thus can achieve a high level of scalability and robustness. A P2P overlay network can be easily applicable to MMGs since players are likely to contribute their communication and computation resources for interesting game-play.

In this paper, we propose a fully-distributed P2P architecture for MMG applications. In a generalized game, players can be conceptually seen as coordinate points on a 2D plane, each with a visibility range called Area of Interest (AOI). In this context, the key issue in P2P gaming is then to define, for a given peer, the set of all of its neighboring peers as well as all objects of the game world within its AOI. In our approach, the game world is divided into zones using Voronoi algorithm [1]. Each peer (player) is responsible for a zone in which it is located, and disseminates game state changes caused by its movements or state changes of objects in its local zone to other concerned peers (i.e., to peers whose AOI cover these changes).

The rest of the paper is organized as follows. Section 2 defines the underlying system model and gives some background on the Voronoi diagram. Section 3 details our architecture and describes our solution in the presence of players' position changes. Section 4 provides a qualitative discussion of our solution. Section 5 presents related works. Finally, section 6 concludes this work and points out some of our future perspectives.

II. SYSTEM MODEL

A. Voronoi Diagram

Given a set S of points in the 2D plane, the Voronoi diagram for S is the partition of the plane which associates a region $V(p)$ with each point p of S in such a way that all points in $V(p)$ are closer to p than any other point of S. Formally, we define $V(p)$ as follows:

$$V(p) = \left\{ x \in R^2 \,\middle|\, |x-p| \le |x-q| \quad \forall q \in S-p \right\}$$

Here, p is called the center point of the region $V(p)$.

B. Voronoi-Based Gaming

The game world is made up of immutable objects, mutable objects and characters controlled by players. We assume that all immutable objects' data are installed as part of the game software on each peer.

We partition the game world into zones based on the Voronoi algorithm [1]. The peer whose position coincides with the center point of a zone is responsible of that zone, following the same region approach as in Voronoi diagram. In a zone, there can also be mutable objects. Each peer is only interested in the activities happening within its AOI (Fig. 1). In our basic model, we assume that peers have the same size

This work is supported by the "Mad Games" project of the ANR RIAM program.

1-4244-1457-1/08/$25.00 © IEEE

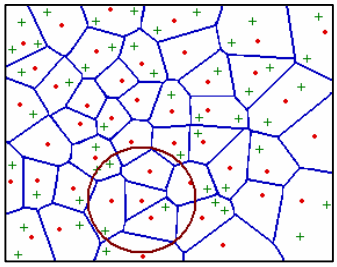

Figure 1. Voronoi diagram.
Lines define boundaries for regions.
Dots indicate center points (peers). Plus (+) represent objects.
The large circle is the AOI boundary for the center peer.

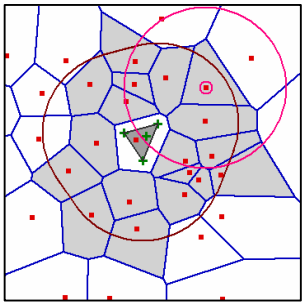

Figure 2. Polygon and its visibility region.
Dark-gray area indicates *polygon*. Convex curve surrounding *polygon* indicates *polygon's visibility region*. Circle shows the AOI of a peer.

of AOI and the radius of AOI is determined in an application-specific manner.

Each peer, say p, keeps a list of other peers that need to be informed about the game state changes occurring in its responsibility zone, and which are visible to these peers. To do so, peer p sends at regular time intervals, say t, the necessary and relevant update information to each of the peers in the list.

When peer p moves (i.e., changes its position), it informs the related peers in its 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, the list of objects that peer p is responsible of might change due to its movement. If so, p gives up the responsibility of objects now lying in the region of another peer and takes over the responsibility of objects now part of its new region. Then, p reconstitutes its peer list and sends relevant information to related peers in the list.

In the following section, we explain in detail the peer list reconstruction issue, and the way we maintain consistency in the face of peers' movements.

III. DATA MANAGEMENT

A. Definitions

Inside the zone owned by peer p, we first define a minimum convex polygon $P(p)$ that contains all objects within p's AOI as well as the responsible peer's position, as shown in Fig. 2.

If there is no object within the zone, then the polygon becomes a single point that represents the position of the responsible peer, namely the center point of the region.

We then define another region, which we call *polygon's visibility region*, which is based on the polygon defined above. If the radius of AOI in the world is r, then for a zone with a responsible peer p and polygon $P(p)$, the definition of the *polygon's visibility region* $VR(p)$ can be given as follows:

$$VR(p) = \left\{ x \in R^2 \,\middle|\, |x-y| \le r \quad \forall y \in P(p) \right\}$$

In Fig. 2, for a given peer p, the *polygon* and the *polygon's visibility region* are respectively represented by the dark-gray area and the convex curve surrounding the *polygon*. The peers within the *polygon's visibility region* (whose zones are colored light-gray) are interested in (part of) the updates caused by the state changes of peer p or the objects in the *polygon*. To this purpose, peer p keeps a list of peers who are located in the *polygon's visibility region*. At every time interval t, peer p informs each peer q in the list about any state changes occurring in the *polygon* if these changes happen within the AOI of peer q.

In this composition, there are two different types of neighborhood relation between peers. For a given peer p, a peer q is called *direct neighbor* if there is a common edge between the zones of p and q. Besides, a peer r is called *boundary neighbor* if (1) r lies inside the *polygon visibility region* of p, and (2) at least one direct neighbor of r lies outside the *polygon visibility region* of p. Note that any peer whose zone has a common edge with p's zone and who has at least one direct neighbor lying outside the $VR(p)$ is both *direct* and *boundary neighbor* of p (Fig. 3). These neighborhood relations are significant for consistency maintenance in the face of peers' position change. This issue is explained in full detail in the following section.

B. Peer's Movement

The list of actions to take in the face of a peer p's movement is defined as follows:

1) p informs the related peers in its peer list about its position change. Towards this end, p first informs its *direct neighbors*. As a consequence of p's position change, *direct neighbors* have to recalculate the boundaries of their zones. 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*. Second, p informs all the peers whose AOI contains p's position (i.e., peers to whom p is visible). Third, p removes the peers from its peer list that are no longer in its *polygon visibility region* if they are not *direct neighbors*. Finally, *boundary neighbors* are asked to check if any of their *direct neighbors* has now become p's direct neighbor.

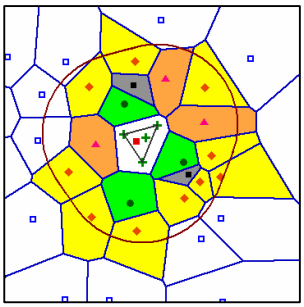

Figure 3. Neighborhood relations.
(●) are *direct neighbors*. (◆) are *boundary neighbors*.
(▲) are both *direct* and *boundary neighbors*.

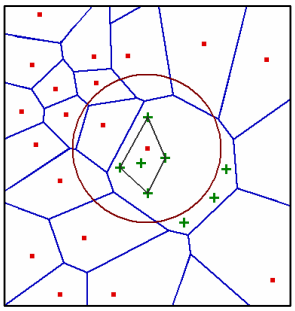

Figure 4. A peer's managed objects.

2) *p* recalculates the boundaries of its zone.

3) Since the boundaries of the zone are changed, some objects may now lie in the zone of any *direct neighbor*. If there is any object which is not lying in *p*'s zone 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 zone the object has newly entered adds the object in its object list and becomes responsible for the dissemination of the object's updates to other relevant peers.

4) Reciprocally to (3), *p* is informed, if needed, by its *direct neighbors* about any newly entered objects into its zone. Consequently, *p* adds these objects in its object list and takes over the responsibility of disseminating the objects' updates to other relevant peers.

5) *p* examines if the *polygon* has changed. If so, *p* recalculates the *polygon* as well as the *polygon's visibility region.*

6) *p* updates its peer list. Towards this end, *p* removes from the peer list all the peers that are no longer in the *VR*(*p*), if they are not *direct neighbors*. Moreover, to detect newly entered peers into *VR*(*p*), *p* asks the *boundary neighbors* to check if any of their *direct neighbors* has entered into *VR*(*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. Finally, *p* sends relevant information to related peers in the peer list.

IV. DISCUSSION

In this section, we make a qualitative discussion of our solution.

Our architecture is fully-distributed since all peers in the system have equal roles and responsibilities in all aspects. No peer has any superiority over another peer in the system.

Our solution is scalable and bandwidth-efficient. Peers have a limited number of connections to other peers, independently from the scale of the system.

The game state consistency takes place in a low-latency manner. Each peer directly connects to the peers which it needs to inform about updates. Therefore, the updates are sent to each related peer in only one communication message.

At first glance, the peers having a large responsibility zone can be seen as overloaded as compared to the peers having a small responsibility zone. However, as previously indicated in the *polygon* definition, peers send the updates of the objects within their AOI (Fig. 4). This is because, for an object *o* in the system, if peer *p* is responsible for the dissemination of *o*'s updates, then *p* is the closest peer to *o* in the game world. Subsequently, if *o* is not visible to *p*, then *o* cannot be visible to any peer in the game world. Consequently, peers are responsible of updates dissemination of the objects within their AOI, regardless of the size of their responsibility region.

It is worth noting that our solution also handles the case of moving objects. Indeed, objects in a game can also change position in the virtual world. If it is the case, then the peer managing the moving object is responsible of the dissemination of this information to all other peers to whom the moving object is visible. If the object's position change induces a change of the region in which the object lies, then the object's responsible peer becomes the peer owning the region in which the object now lies.

Finally, and as a proof of concept, we have simulated a simple game application and tested the correctness of our solution in the presence of moving peers and objects.

V. RELATED WORKS

Although multiplayer networked gaming is becoming an important and popular application, works that exploit the peer-to-peer technology in this context are still at their beginning.

In [2], 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, thus 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 distributed hash tables. This solution can clearly suffer from classical performance, scalability, and robustness problems intrinsic to client/server architectures

especially in dynamic games where all objects and nodes might end up in a single region managed by a single server, and thus does not represent a good candidate for fully distributed peer-to-peer gaming.

In [4], 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 [2], this solution also suffers from the same problems related to client/server architectures.

The work initially presented in [3] and later extended in [9] is most similar to ours in the sense that the authors discuss a Voronoi-based partitioning of space in the context of networked virtual environments. This work, however, 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. This is exactly the major contribution of our paper.

VI. Conclusions and Future Work

In this paper, we have proposed a fully-distributed architecture to support massively multiplayer games in a peer-to-peer overlay network. By exploiting interest management characteristics of games, we designed a scalable mechanism to distribute game states over all the peers participating in the game. The general idea of our solution is that each peer is responsible of the dissemination of game state changes taking place in its surrounding to other peers interested in these state changes. The partitioning of the virtual game world, needed to associate a responsibility zone with each peer, is based on Voronoi algorithm. Each object in the game world is managed by the closest peer. Objects' updates and peer positions' updates are sent to each nearby peer in only one communication message, making our solution highly efficient and scalable.

As a future work, we intend to evaluate our solution in a real game application on a P2P overlay network. It would be also interesting to consider different parameters, such as different object visibility levels depending on object's size, peers' heterogeneity for load balancing purposes when peers have different capacities, and peers' energy levels so as to maximize the network life.

Acknowledgment

We are thankful to Baver Okutmustur for his precious help and valuable feedbacks. We are also thankful to Guray Erus for his interesting discussions and insightful comments that helped improve the paper.

References

[1] F. Aurenhammer, "Voronoi diagrams-a survey of a fundamental geometric data structure," ACM Computing Surveys (CSUR), vol. 23(3), pp. 345-405, September 1991.

[2] B. Knutsson, H. Lu, W. Xu, and B. Hopkins, "Peer-to-peer support for massively multiplayer games," IEEE Infocom, March 2004.

[3] S. Y. Hu and G. M. Liao, "Scalable peer-to-peer networked virtual environment," 3rd ACM SIGCOMM workshop on Network and system support for games (NetGames), pp. 129-133, August 2004.

[4] A. Yu and S. T. Vuong, "MOPAR: A mobile peer-to-peer overlay architecture for interest management of massively multiplayer online games," International workshop on Network and operating system support for digital audio and video (NOSSDAV), pp. 99-104, June 2005.

[5] T. Iimura, H. Hazeyama, and Y. Kadobayashi, "Zoned federation of game servers: A peer-to-peer approach to scalable multi-player online games," 3rd ACM SIGCOMM workshop on Network and system support for games (NetGames), pp. 116-120, August 2004.

[6] Y. Kawahara, H. Morikawa, and T. Aoyama, "A peer-to-peer message exchange scheme for large scale networked virtual environments," 8th IEEE International Conference on Communication Systems (ICCS), pp. 957-961, November 2002.

[7] J. Keller and G. Simon, "Solipsis: A massively multi-participant virtual world," International Conference on Parallel and Distributed Techniques and Applications (PDPTA), pp. 262-268, 2003.

[8] A. Bharambe, J. Pang, and S. Seshan, "Colyseus: A distributed architecture for multiplayer games," ACM/USENIX NSDI, May 2006.

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