nSense: Decentralized Interest Management in Higher Dimensions through Mutual Notification

Robert Rehner Maribel Zamorano Castro Alejandro Buchmann
Databases and Distributed Systems, Technische Universität Darmstadt, Darmstadt, Germany
{rehner, zamorano, buchmann}@dvs.tu-darmstadt.de

***Abstract*—Interest management is a key component of decentralized multiplayer gaming overlays. Current approaches are limited to two, sometimes three spatial dimensions. Considering games that use game features like portals and unit-type visibility, this limitation either forces the game designer to omit such features or it causes increased latency and bandwidth for the player. In this work we present and evaluate *nSense*, a decentralized interest management scheme that can be used for an arbitrary number of dimensions. It uses mutual notification so it only needs to connect to players with close proximity. Therefore it is highly scalable and suitable for massive multiplayer online games.**

I. Introduction

In Multiplayer Online Games, *Interest Management* (IM) is the mechanism that determines which of a player's actions need to be transmitted to which other players. IM can be performed by a central instance, which then inevitable becomes a scalability bottleneck and a single point of failure. Proposals for decentralized IM exist [1], but are limited to two, sometimes three, spatial dimensions. In the light of games like *Portal* or *Miegakure* that already make use of more than three dimensions, this limitation appears anachronistic. In this paper we present and evaluate a fully decentralized Interest Management scheme called *nSense*, which supports an arbitrary number of dimensions. Our approach is based on mutual notification of the peers and therefore also solves the neighbor discovery problem which is another challenge of decentralized gaming overlays.

Due to the support of more than the common two to three dimensions directly in the network overlay, corner cases of previous approaches like *portals* or *warp holes* are eliminated. Having these corner cases introduces delays for finding the nodes at the other end of the portal (by gossiping, DHT lookups, etc.) and establishing connections to them before executing the jump. In nSense these corner cases no longer exist for the network overlay: moving through a portal can now be modeled as a regular movement in the d-dimensional space that needs no special treatment on the network overlay. Only the graphics, which are calculated locally, need to be aware of the portal to map the d-dimensional space correctly to 3D and render some special effects for portal. Further use cases for more dimensions include unit type or team visibility, or mapping node capabilities like bandwidth and latency as additional dimension to use them for load balancing [2].

II. State of the Art

The approaches for handling MMOGs can be categorized as being either centralized, decentralized or hybrid. Centralized solutions have the typical drawback that the central instance is the single point of failure and will eventually become a bottleneck. Therefore some approaches use multiple servers that communicate with each other in a peer-to-peer (P2P) fashion but maintain the centralized facade for the players. On the other side of the spectrum are the decentralized approaches. Buyukkaya et al. [1] provide a overview of the different proposed approaches. Despite their diversity, they all have one limitation in common: their limited dimensionality. Most approaches are suitable for two-dimensional games; some can deal with three dimensions. But none of them can handle more than three dimensions.

Voronoi-based approaches could be extended to more dimensions, but only theoretical considerations exist. An example based on a voronoi-approach is given by Almashor and Khalil [2]: they use a 3D voronoi-based overlay for a 2D game. The third dimension represents the bandwidth of a peer and is used for load balancing. The increased computational complexity associated with Voronoi computations in higher dimensions has to be taken into account and may lead to a degraded performance.

A simple idea is to ignore the additional dimensions and use only the ones supported by the overlay. This results in a change of the AOI shape and size as shown in Figure 1, i.e., if the original AOI in a 3D game was ball-shaped and one maps this to a 2D overlay by omitting one dimension the AOI becomes a cylinder. This is a severe problem because the AOI becomes bigger and players far away but directly above or below the player would be considered within the AOI.

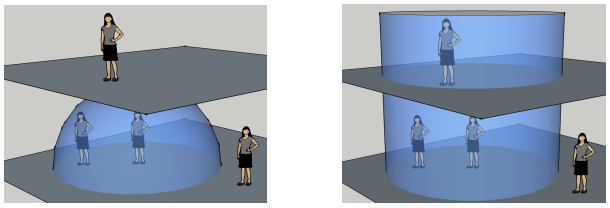

Figure 1. Mapping a 3D spherical AOI to a 2D overlay yields a cylindrical AOI where more players than necessary are seen and therefore network traffic is increased. The effect worsens with more dimensions.

Another approach would be to use space-filling curves, like the Z-order curve or Hilbert curve [3]. Space-filling curves preserve locality to a certain degree and points that are adjacent in the d-dimensional space are mapped close together to the 1-dimensional space. However, no matter what curve is used, the locality inevitably becomes very low in certain situations [3].

Therefore, we argue that multiple dimensions should be directly supported by the overlay to avoid these problems.

III. nSense

One important aspect of an IM is the decision which players can see and interact with each other. In our approach we assume that each player has an Area of Interest (AOI) surrounding her. Everyone within his AOI is visible and can be interacted with; everyone outside of the AOI can not be seen

978-1-4799-6882-4/14/$31.00

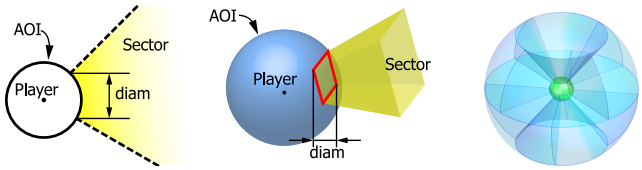

Figure 2. Dividing the space into sectors; *diam* specifies the diameter of the exemplary sector. Left: In 2D we obtain a triangle-shaped sector. Middle: In 3D we obtain a pyramid-shaped sector. Right: 3D sphere with 10 sectors.

by the player. The AOI could have various shapes depending on the underling game [4]. In this work we assume, without loss of generality, that the AOI is a sphere around the player.

The challenge is to know when someone enters the AOI. For that we use a mutual notification scheme based on the idea of pSense [5], which only supports two dimensions. Seen from one peer the other peers are divided in three groups: within the AOI, sensors, and other players that are unknown. Players within AOI are known and a connection is maintained to them; communication (e.g., position updates or player actions) occurs regularly either by direct connection or via a multicast scheme [6]. The outside of the AOI is divided in several *sectors*, where we choose a *sensor node* within each sector. They inform the player about incoming other players as well as about other players that would be a better sensor than themselves. The ideal sensor node for a sector is a node within this sector that is closest to the border but still outside of the AOI. It can happen that there is no node in a given sector and outside the AOI. In this case the node within the AOI and within the sector which is closest to the AOI border is chosen. This setup achieves high scalability since the number of connections is only dependent on the local player density and independent of the total number of players.

The problem of dividing the region outside of the AOI in sectors can be reduced to the problem of dividing the surface of a d-dimensional sphere in areas. The sector is then defined by the space that can be 'seen' by looking through this area from the center of the sphere. The first idea for this approach was presented earlier [7] and is illustrated in Figure 2.

For the selection of the partitioning method we define two basic goals: minimizing the number of sectors n and the diameter of the sectors. (1) Since the node needs to communicate regularly with each sensor node, it is desirable to keep the number of sensors as small as possible. (2) The diameter of the sector should be small, or else a player could 'sneak' by the sensor node and enter the AOI without being seen. From this we can infer that all sectors should have the same size: else it would mean that our division is sub-optimal and has more sectors than necessary.

A. Sphere Partitioning Algorithm

Since position changes occur constantly in multiplayer games, we need a sphere partitioning algorithm that satisfies the above requirements and is simple to compute. For this work we use the method presented by Leopardi [8]. Its geometrically simple sector shapes lead to a simple calculation scheme where only some angles need to be computed to determine the sector of a given point (i.e., a player position). The so-called *recursive zonal equal area partition* algorithm, $\mathrm{EQ}(d, N)$, takes the dimensionality d and the number of desired partitions N as input. It partitions a d-dimensional sphere in collars with subdivisions; the two polar caps can be treated as collars with

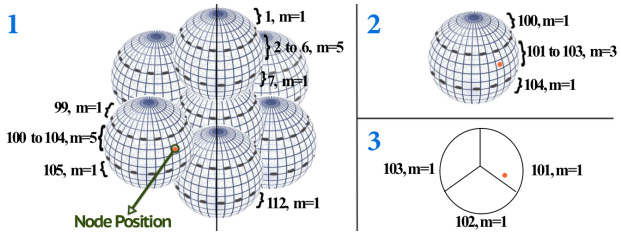

Figure 3. An example of how to obtain the sector ID of a point in 4D space. We start with a sphere with 112 sectors and determine the collar for the point. Next we recursively use the same method on the collar which now has only three dimensions.

no subdivision. The output are the angles corresponding to the collars, the number of collars n and the number of subdivisions for each collar $(m_1, ..., m_n)$, such that $N = 2 + \sum_{i=1}^{n} m_i$. Figure 2 (right) shows an example of a 3D sphere partitioned in 10 sectors. The actual subdivision of collar i is then obtained by applying $\mathrm{EQ}(d-1, m_i)$ recursively until the collar consists of only one partition. We pre-compute the values for a wide range of values for d and N at build time. Since for each configuration, i.e., dimension d and sector number N, we only need to save $\mathcal{O}(d * n)$ values.

With the help of the pre-computed values we can easily determine the sector of another player, i.e., the sector of a given point $\vec{p} = (p_1, \ldots, p_d)$. The algorithm works as follows: Let $\vec{e}_1, \ldots \vec{e}_d$ be the basis vectors of the d-dimensional space.

1) determine the angle α between $\vec{p}$ and $\vec{e}_i$ for $i = 1$.
2) look-up to which collar α belongs.
3) if the collar consists of only one region ($m = 1$), we are *done* and output the sector of the point.
4) for collars with several regions ($m > 1$), we recursively repeat the algorithm with $i := i + 1$.

The algorithm is illustrated in Figure 3. It computes at most $d-1$ angles between p and the base vectors. Which leads to at most $log(n) * (d-1)$ comparisons to the pre-computed angles.

B. Communication protocol

This section describes the *nSense* protocol messages. For simplicity, we assume that a reliable transport protocol is used.

New connection Each node can initialize a connection by contacting another node via its IP:Port combination.

Quit When a node leaves the AOI the connection will be terminated after a threshold time by sending a *quit* message.

Position updates to nodes within the AOI and to all sensors.

Game actions Game event dissemination is not a part of nSense, but it can be done similarly to position updates.

Join A node joins by contacting an arbitrary peer from the overlay. This peer determines a joining position in his vicinity and sends the node the contact data of the neighbors.

Sensor request If a node determines that a node would be a suitable sensor it sends a sensor request containing the sector number for which the node will be the new sensor.

Sensor quit The receiver is removed as a sensor.

Sensor suggestion The sensor s of player p usually knows other players outside of the AOI of p. If s determines that another player outside of the AOI of p would be a better sensor node than himself, s sends a *sensor suggestion* to p, i.e., the contact data of s'. Now, p will connect s' and follow its normal sensor selection algorithm but with a better knowledge of its surroundings.

Node suggestion A sensor can also introduce the player to new nodes that are about to enter the player's AOI.

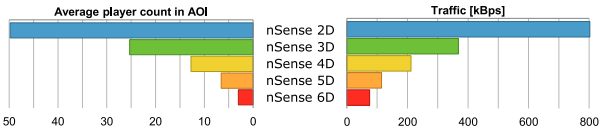

Figure 4. A 6D workload handled by overlays with different dimensionalites.

IV. Evaluation

Our system simulates a multi-dimensional game where peers, i.e., players, can move around. All players have the same AOI size: in 2D this yields a circular AOI, in 3D a sphere and a hypersphere in more than three dimensions. The requirements of our system are based on a typical multiplayer game: accurate detection and knowledge about all players within the AOI, timely position updates of all players within the AOI, and keeping network traffic as low as possible.

A. Simulation Environment and Workload

We use the discrete event network simulator PeerfactSim.KOM [9] to simulate sessions of 5 minutes with 300 players. For this work we extended its workload generator so that players move in multiple dimensions. We use a *random point of interest* movement model, because it is reproducible, scalable, and represents human behavior better than random way point models [10], As performance metrics we use: (1) Bandwidth consumption: obviously, in our setup the used bandwidth depends mostly on the amount of players within the AOI and the number of sensor nodes connected. (2) Precision: to determine if the known players should be known to a player, i.e., are within his AOI. (3) Recall: to determine if all players within the AOI are known.

B. Simulation Results

First, we compared nSense-2D to pSense to establish that our algorithm yields comparable results in 2D. When pSense is used with a 3D workload its performance drops due to the effects discussed in Section II, whereas nSense-3D performs well. Next we simulate a six-dimensional world. Figure 4 (left) shows the average number of players within the AOI. The expected number would be only three players and is achieved by nSense-6D. When we map the 6D space to less and less dimensions in the overlay the number increases up to 50 players within the AOI. The consequence of more players within the AOI is illustrated in Figure 4 (right): the traffic increases. We disabled the per-node traffic limit to obtain these results. In a real world situation most upload capacities found in home broadband connections would be saturated.

We also simulated nSense with different numbers of sectors and observe that 8 sectors yields the best results. This is due to two facts. First, more sectors increase the bandwidth consumption which increases the probability for a node to be overloaded. Secondly, the chance that a sector remains without a sensor is increased. In such a situation it can happen that a node "sneaks" into the AOI without being seen on time.

Our results also indicate that bandwidth consumption could be further reduced by reducing the number of sectors, while maintaining good values of recall and precision.

V. Conclusion and Future Work

Today's decentralized multiplayer gaming overlays are limited in their dimensionality to two or three spatial dimensions. Elements that make more dimensions useful are already used in games. For example portals or unit-type visibility have already been used in games decades ago. These elements become a challenge when we move away from a centralized architecture. Then a portal results in a disruption of the localized overlay and causes increased latency for the players. Additional dimensions can also be used for other purposes like mapping the bandwidth of a node to a dimension and use it for load balancing.

We have presented *nSense*, a decentralized interest management scheme that is suitable for games with an arbitrary number of dimensions. It uses localized communication only and is therefore highly scalable. The overlay is built by using direct communication with players within the *area of interest* (AOI) and maintaining additional connections to so-called *sensor nodes*. These nodes are guards for a certain part of the world outside of the AOI and inform the player about other players that are about to enter its AOI. They can also suggest other nodes as sensor nodes if they are more suitable.

We have implemented and evaluated our approach and shown that it outperforms algorithms that just ignore additional dimensions.The main benefit is that we maintain connections only to players that are within the AOI in a multi-dimensional world. Current approaches with a limited dimensionality have to maintain up to an order of magnitude more connections and will therefore encounter bandwidth saturation.

As further work we plan to implement our approach in a multiplayer game that makes use of more dimensions. It would also be interesting to evaluate different AOI shapes and AOIs with dynamic changing sizes during the gameplay. We also plan on looking into adapting the number of sectors dynamically for each player depending on the local player density. If players within the AOI would participate in the mutual notification it could eliminate the need for sensor nodes in crowded areas and therefore save bandwidth.

Acknowledgment

The Authors would like to thank *Michael Stein* and *Björn Richerzhagen* for their help with PeerfactSim.KOM. This work has been co-funded by the DFG as part of CRC 1053 *MAKI* and the GRK 1343 *Topology of Technology*.

References

[1] E. Buyukkaya, M. Abdallah, and G. Simon, "A survey of peer-to-peer overlay approaches for networked virtual environments," *Peer-to-Peer Networking and Applications*, pp. 1–25, Sep. 2013.

[2] M. Almashor and I. Khalil, "Load-Balancing Properties of 3D Voronoi Diagrams in Peer-to-Peer Virtual Environments," *IEEE Parallel and Distributed Systems*, pp. 839–844, Dec. 2010.

[3] P. Ganesan, B. Yang, and H. Garcia-Molina, "One Torus to Rule them All: Multi-dimensional Queries in P2P Systems," in *WebDB'04*. New York, New York, USA: ACM Press, Jun. 2004, pp. 19–24.

[4] S.-Y. Hu, "Spatial Publish Subscribe," in *IEEE Massively Multiuser Virtual Environment (MMVE'09)*, 2009.

[5] A. Schmieg, M. Stieler, S. Jeckel, P. Kabus, B. Kemme, and A. P. Buchmann, "pSense-Maintaining a Dynamic Localized Peer-to-Peer Structure for Position Based Multicast in Games," in *Peer-to-Peer Computing (P2P'08)*, 2008, pp. 247–256.

[6] M. Lehn, R. Rehner, and A. Buchmann, "Distributed Optimization of Event Dissemination Exploiting Interest Clustering," in *IEEE Local Computer Networks*, 2013.

[7] R. Rehner, M. Lehn, and A. Buchmann, "nSense: Interest Management in Higher Dimensions," in *Peer-to-Peer Computing (P2P'13)*, 2013.

[8] P. Leopardi, "A partition of the unit sphere into regions of equal area and small diameter," *ETNA*, vol. 25, pp. 309–327, 2006.

[9] D. Stingl, C. Gross, J. Ruckert, L. Nobach, A. Kovacevic, and R. Steinmetz, "PeerfactSim.KOM: A simulation framework for Peer-to-Peer systems," in *HPCS*. IEEE, 2011, pp. 577–584.

[10] T. Triebel, M. Lehn, R. Rehner, B. Guthier, S. Kopf, and W. Effelsberg, "Generation of Synthetic Workloads for Multiplayer Online Gaming Benchmarks," in *NetGames'12*, vol. 4. IEEE, Nov. 2012.