Clustering Players for Load Balancing in Virtual Worlds

Simon Rieche, Klaus Wehrle
Distributed Systems Group
RWTH Aachen University
{rieche,wehrle}@cs.rwth-aachen.de

Marc Fouquet, Heiko Niedermayer, Timo Teifel, Georg Carle
Computer Networks and Internet
University of Tübingen
{fouquet,niedermayer,carle}@informatik.uni-tuebingen.de

***Abstract*—Massively Multiplayer Online Games (MMOGs) have become increasingly popular in the last years. So far the distribution of load, caused by the players in these games, is not distributed dynamically. After the launch of a new game, the introduction of new content, during special ingame events, or also during normal operations, players tend to concentrate in certain regions of the game worlds and cause overload conditions. Therefore we propose the use of structured P2P technology for the server infrastructure of the MMOGs to improve the reliability and scalability. Previous work segmented the game work into rectangular areas; however this approach often split a group of players to different servers, causing additional overhead.**

This work presents a cluster-based Peer-to-Peer approach, which can be used for load balancing in MMOGs or in other virtual worlds. The system is able to dynamically adapt to the current state of the game and handle uneven distributions of the players in the game world. We show through simulation, also with traces from real online games, that the cluster-based approach performs better than the previous P2P-based systems, which split the world in rectangular areas.

I. Introduction

Computer games have changed in the last years due to the constantly rising speed and decreasing costs of internet connections. Ever more games offer the possibility, to play with other players together over the internet. Particularly in Massively Multiplayer Online Role-Playing Game (MMORPGs) a virtual world is created, in which thousands of players "live". For the players the shared virtual world offers the possibility that they can talk with other human players, build groups, friendships and fight together.

Traditionally, a cluster of servers contains one virtual world of a Massively Multiplayer Online Game (MMOG). But such an infrastructure is inflexible and error-prone. One would rather like to have a system that allows disconnecting a node at runtime while others take over its tasks. Also server-based MMOGs can have performance problems if players concentrate in parts of the game world or if some worlds are overpopulated. Thus, there is a need for load balancing mechanisms, which Peer-to-Peer (P2P) systems quite naturally support.

In previous work we showed how such a game can be hosted on a P2P-based infrastructure [1]. By using the structured P2P System Content Addressable Network (CAN) [2] as a basis, we split the game world into disjunctive zones and distribute them on different nodes of the P2P network. This happens automatically in such a way that all existing servers have nearly the same load. But the load balancing separates the world in a quad tree-like fashion and does not take the structure of the map into account, which may lead to many changes of players between the servers. In this case the old server has to transfer all the state information of the player to the new server, which is now responsible for this person. So a main task is to minimize the handovers of players from one server to another. Another problem are players who are located close to the border of the area that one server controls. These players can see parts of the game world on the other side of the border, therefore they need to be informed about all updates of dynamic content, i.e. player movements that happen there.

In MMORPGs it can be frequently observed that players form groups and walk over the map together. This particularly becomes problematic, if a whole group of players moves over a border and thus a large number of players have to be transferred to the new server. Thus we propose a P2P infrastructure for MMOGs which takes this behavior into account.

The distribution of load on the existing servers is thereby not done by dividing the map into different areas, but via dividing the players into clusters. Each server receives a group of players, who are close together. This group is one cluster. If the group moves on the map, the responsible area of the server moves with this group. If individual players depart from the group, they are handed over to another group as soon as they are closer to the new group than to the previous one. Figure 1 shows as an example several clusters in a virtual world.

The rest of this paper is organized as follows: First we discuss related work in Section II. Section III shows our

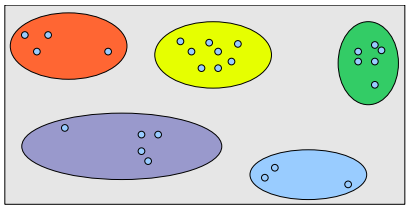

Fig. 1. Several clusters on the map of the virtual world.

approach to use structured P2P Systems for MMOGs, Section IV how the clusters are build, and Section V how to find clusters in the neighborhood of the map. In Section VI the load balancing of the cluster is described. Section VII describes the evaluation, including simulations that use player traces from a real MMOG, and finally, Section VIII provides conclusions.

II. Related Work

Some efforts have been undertaken to design a MMOG on a P2P basis, with the server tasks being shared among the player's PCs. In [3] and [4] the game world is divided into zones. Then some peers become zone owners that take responsibility for computing the server tasks for such a zone. While this approach is fascinating on one hand, it suffers from a number of practical problems. Players constantly connect to and disconnect from the game, often without warning if the PC crashes or suddenly gets disconnected from the network. This means that always backup machines are needed to replace disconnected zone owners. Allowing player's computers to calculate parts of the game mechanics makes it harder to avoid cheating. Another problem is that persistent player data, for example the progress of the player's characters in a role-playing game, need to be saved in a way that makes sure that no information is lost, as players may have invested a lot of work into the game. At the same time one has to prevent players from cheating by modifying data, which would be possible if it was stored on the player's local hard drives. This suggests that some kind of infrastructure provided by the game manufacturer would still be needed; a full P2P approach does not appear practical, so we focus on a hybrid P2P approach. Another challenge is the low upstream bandwidth of most current Internet connections, probably only peers with a good connectivity could be considered for becoming zone owners.

Solipsis [5] uses a different approach, as it tries to build a pure P2P network-based on the neighbourship relations of the player's avatars. Each peer has direct connections to all other peers that are in visible range of the player's avatar. There is a real implementation of Solipsis, however currently it is little more than a distributed chat client. If one wanted to make it a "real" MMOG, one would be confronted with the same problems as described above. It is especially difficult to make such an approach cheat-proof [6].

In our previous work [1] a structured P2P technology is used for the organization of the underlying infrastructure and thus for the reduction of downtimes in MMOGs. By using a CAN-based approach we split the game world in disjunctive zones and distribute them on different nodes in the P2P network. Thus, we get the possibility to dynamically connect and disconnect machines to and from the peer-cluster and to load-balance the game according to the actions of the players.

The P2P technology makes it possible to run multiple game worlds on a pool of servers. The physical location of these servers is of minor importance, they do not have to run in the same data center. Placing the servers of a single game world at different locations introduces additional overhead; however this may be justified by enhanced reliability or maybe by an improved locality. Even if one whole location should be disconnected from the network, the servers at other locations could take over seamlessly and without loss of data.

But it can be observed that players act in groups in most online games. Since those players can see all players in the same group on the virtual map, many messages have to be sent over the server to inform where the other players are. If there is a border between these players in the map the additional data has to be exchanged between the two servers, which are responsible for the different areas.

Many algorithms exist for load balancing in structured P2P systems [7]–[9]. However, most of these systems are not applicable for online games. Our approach is based on the virtual server (VS) approach [7] where multiple partitions of a Distributed Hash Table's (DHT) address space are managed in one node. Thus, one physical node may act as several independent logical nodes. So each VS will be considered by the underlying DHT as an independent node.

III. Cluster-based P2P Infrastructure Support for MMOGs

In our approach, the game world is defined by a map, which is managed by servers. Each server is responsible for the region, where the players of its group are located. Conceptually areas without players don't need a responsible server. Non-player characters and other objects with a state can be handled as players. So the game world is distributed on an infrastructure of different peers or servers. This infrastructure is not necessarily located in a single data center. However in most cases it makes sense to locate the peers that are responsible for adjacent regions in the same network to minimize delay and costs for internal traffic between the peers. But our approach could also be used for a super-peer network in a more distributed setting for more delay-tolerant games.

As in the virtual server (VS) approach multiple peers can be run on one physical server to better distribute the load. Since no central server should be used for the whole game, the movement of players from one cluster to another is done only by the two participating (virtual) servers. Each VS knows the coordinates of the players of its own group and the positions of some players of neighbouring groups which can be seen by the players of its group. Neighbouring virtual servers periodically exchange this information, to make sure that the positions of all players are rendered correctly on the player's computer.

IV. Building Clusters

For building the clusters, we measure the distance between the players. Simply measuring each player's distance to the centroid of a cluster (cf. Figure 2) leads to a suboptimal solution since groups may be split in the middle of two virtual servers. So this approach is not flexible enough to handle large or deformed clusters.

Figure 3 shows how the mapping of players to groups should be determined. Players belong to the same group, if they have a small distance to each other. The form of the group can be arbitrary, since the distance is not computed to a central

point, but to each player's neighbours. The required minimum distance to separate two players into different clusters is variable, so the size of the groups can be changed e.g. when adding new nodes. Therefore this parameter can also be used for the load distribution. Players, who have the same distance to two large groups, can be taken by any of these groups. A VS is not assigned to a fixed location on the map - as players move around, the VS assignment follows them.

V. Finding Neighbouring Clusters

Since there is no central server, each VS must be able to decide, which other virtual servers of the P2P structure are direct neighbours on the virtual map. Direct neighbours of a VS A are those servers, which have players in their groups, who can move to the players of group A without being transferred to another group of a VS B before arriving at A. In many cases these players are also close together, however there may also bigger areas on the map without any players. A cluster needs connections to all direct neighbours and all clusters are connected in one graph.

It is not sufficient to simply use the distance to a neighbour's centroid to test if two groups are direct neighbours since the groups do not necessarily have to be circular. Also groups, which are far from each other, may be direct neighbours, if no other group is between them.

So in order to test whether a VS A is a direct neighbour to B, the canonical approach would be to compare the positions of all players. For the players, with minimal distance, it must be tested if no players of another VS C lie between them. The test for the neighbourship relation can be optimized, as it is sufficient to test the players that form the convex hulls of the two groups instead of comparing all players.

VI. Load Balancing using Clusters

As mentioned before, our proposed system works with a VS approach. A cluster of players is equivalent to one virtual server. When the load of a physical server exceeds a threshold, for example when too many players are in its clusters, the load is reduced by three mechanisms:

- Moving whole clusters from one node to another.
- Moving one or some players from one cluster to another.
- Splitting a cluster into two parts, and moving one of them to another physical server.

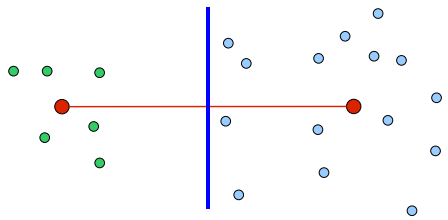

Fig. 2. Building clusters using the distance from the centroid.

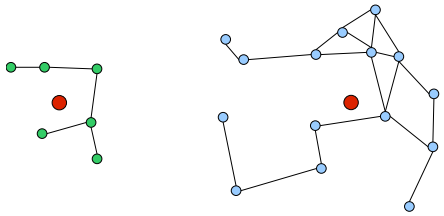

Fig. 3. Building clusters using the distance between players in a group. For illustration the centroids of the groups are also shown.

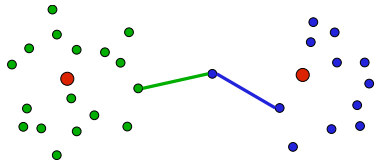

Fig. 4. Moving one player from one cluster to another.

A. Moving Clusters

The virtual server approach [7] is based on the idea of managing multiple partitions of a structured P2P address space in one node. Thus, one physical node may act as several independent logical nodes. Each VS will be considered as an independent node by the underlying structured P2P System. Within a CAN system, one VS is responsible for a zone of the address space, whereas the corresponding physical node may be responsible for several different and independent zones. The basic advantage of this approach is the simplicity of placing and transferring virtual servers among arbitrary nodes. This operation is similar to the standard join or leave procedure in a structured P2P system. Every participating node manages many virtual servers so load can be moved between nodes by moving a whole VS to another node.

B. Moving Players

Another simple operation to balance the load of the clusters is to move some players from one group to another. Figure 4 shows a player in the middle of two clusters. So depending on the load of each cluster the player can be moved to the group with the lower number of players. If some players move away from a cluster together, also all of them can be moved to the new cluster together (cf. Figure 5).

C. Splitting Clusters

Additionally, clusters with too many players can be split and one part can be sent to another server. This is done by

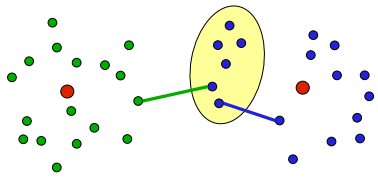

Fig. 5. Moving a part of a group to another cluster.

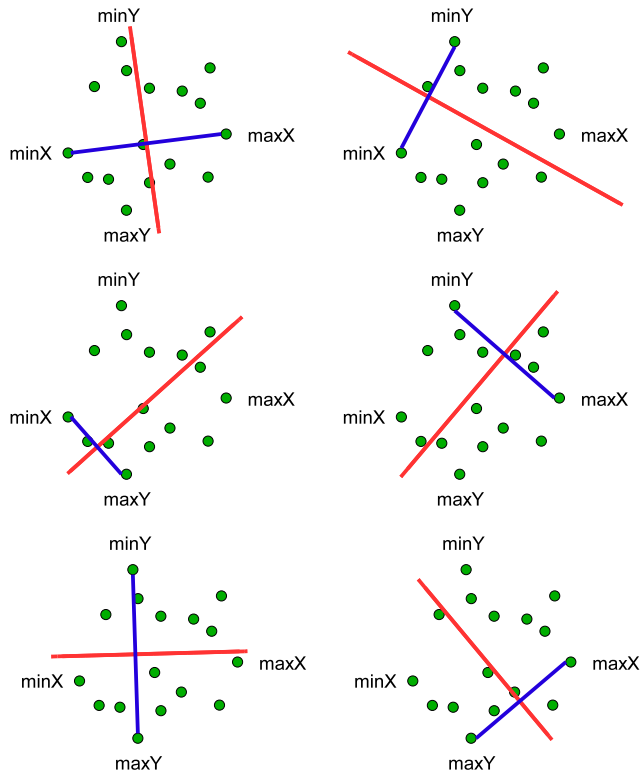

Fig. 6. The virtual server tests six possibilities for splitting a group based on four points. The two points, which are used as starting points to calculate the groups in each round, are connected with a line (blue). The perpendicular bisector of the side drawn in addition isolates the two groups. Each player belongs to the group of the starting point where he is nearer to. So the perpendicular bisector of the side splits the two groups, since after this line the player would be nearer to the other staring point.

first selecting the – usually four – players with minimal and maximal X- and Y-coordinates in the map.

For every one of the six combinations, two players p and q are selected and a split is calculated by simply assigning each other player r either to p's or q's group depending on r's distance to p and q.

At the end the combination of those two nodes p and q is chosen, which has the maximal distance between the two new groups. This is done to minimize the possibility that players will move from one group to the other in the future. Figure 6 shows as an example all six possibilities to split a group.

Also, adjacent clusters with low numbers of players can be merged for a lower number of internal messages.

VII. Evaluation

We focus in this evaluation part, due to space limitations, on the comparison of the cluster-based approach and the CAN-based approach with rectangular areas. In [1] we showed already that a P2P approach for a MMOG is practicable.

A. Traces

We evaluated our approach using a simulation with artificially generated as well as real traces of user-movement:

- **Random walk trace data** is a collection of generic data representing movement of users according to the Random Walk Mobility Model.
- **Random waypoint trace data** has been generated using the Random Waypoint Mobility Model.
- **Freewar trace data** consists of real player-movements traced in the online game Freewar [10] over a period of up to five hours. Approximately 400 players were online in this period of time, but the number of concurrent players varies over time since users join or leave the game. Also the Freewar traces show an extremely uneven distribution of the players over the world with some hotspots in cities.

B. Comparison

We compare the two approaches based on the following criteria using the Omnet++ network simulator:

- The number of ForeignPositionChange (FPC) messages, sent during the simulation. These are sent, whenever a player moves to the peripheral area of his group and can be seen by players of a neighbouring group. A FPC causes the transmission of one message per player and movement (time step) between the neighbouring virtual servers. So the number of FPC messages indicates how well the clusters were separated in the simulation. In a real game this number of FPC messages is one of the most important factors, since many FPC messages between the clusters will delay the response to the users.
- The number of players moved from one VS to another. The counter of handovers is increased with each handover of a single player. When parts of a group are transferred between two virtual servers at once, this counter is increased by the number of moved players. Besides the used bandwidth, the transfer of a player may cause a short delay for the customer. Therefore the number of handovers should be minimized.
- The number of players moved when complete virtual servers are moved between two nodes. This is separately counted in the counter handoversOnMove.

Figure 7 shows the number of handovers of players for a random walk trace. It shows that the cluster-based approach performs better than the CAN-based approach with rectangular areas. Additional, less FPC messages are sent in this simulation (cf. Figure 8). This random walk trace consists of 219,828 movement actions by 500 different players. All graphs are cumulative, i.e. they show the number of messages since the start of the simulation run.

We also simulated a scenario with random waypoint trace data. This trace again consists of 500 players, which move around in the game world for about five hours. Again, the cluster-based method performs better than the CAN-based method. The number of handovers using cluster-based method is 14,889 compared to 28,198 with the CAN-based method. The number of FPC messages is 79,492 for the cluster-based method, compared to 379,392 for the CAN-based method.

In a simulation with the Freewar traces there are less FPC messages sent by the cluster-based method than by the

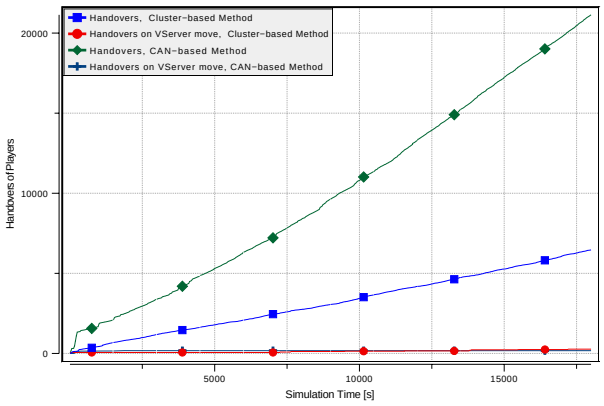

Fig. 7. Comparison of player handovers from one cluster to another with the CAN-based rectangular areas approach and the cluster-based approach with random walk trace data.

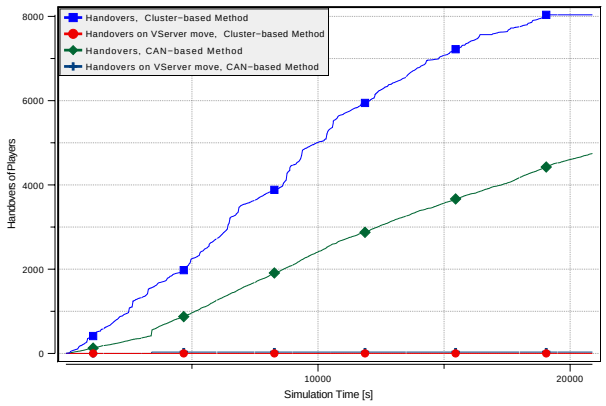

Fig. 9. Comparison of player handovers from one cluster to another with the CAN-based rectangular areas approach and the cluster-based approach with the Freewar trace data.

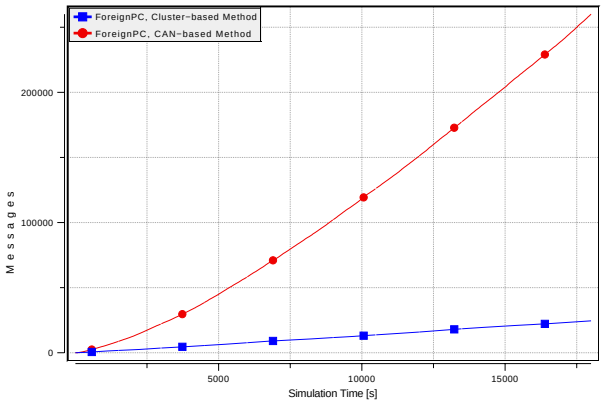

Fig. 8. The number of FPC messages with the CAN-based rectangular areas approach and the cluster-based approach with random walk trace data.

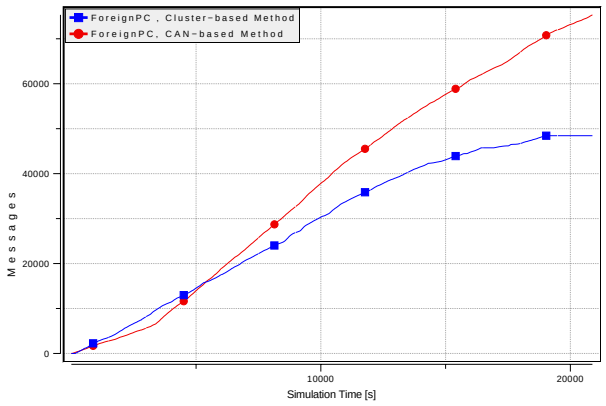

Fig. 10. The number of FPC messages with the CAN-based rectangular areas approach and the cluster-based approach with the Freewar trace data.

CAN-based algorithm (cf. Figure 10). As described in section VII-B, this is one of the main motivations for the cluser-based approach in order to improve the interactive experience for the players. But in the Freewar simulation the cluster-based method (cf. Figure 9 has a worse performance than the CAN-based approach with respect to player handovers. The CAN-based method causes 4,813 handovers and 75,325 FPC messages. The cluster-based method causes 8,038 handovers and 48,441 FPC messages.

So whether our CAN-based or the cluster-based approach is more suitable for a practical game depends on the typical behavior of the players in the game world and on the costs of FPC messages versus player handovers.

VIII. Conclusions

In this paper we use a structured P2P technology for the organization of the infrastructure and thus for the reduction of downtimes in MMOGs. By using a cluster-based approach we split the game world in groups of players and not in rectangular disjunctive zones. The clusters are distributed on different nodes of the P2P network. Thus, we get the possibility to dynamically connect and disconnect machines to and from the peer-cluster and to load-balance the game according to the actions of the players. The evaluation shows a better behavior than the previous CAN-based approach, which created rectangular areas.

References

[1] S. Rieche, K. Wehrle, M. Fouquet, H. Niedermayer, L. Petrak, and G. Carle, "Peer-to-Peer-based Infrastructure Support for MMOGs," in *Proc. of CCNC*, Las Vegas, 2007.

[2] S. Ratnasamy, P. Francis, *et al.*, "A Scalable Content-Addressable Network," in *Proc. of the ACM SIGCOMM*, San Diego, 2001.

[3] T. Iimura *et al.*, "Zoned federation of game servers: a P2P approach to scalable MMOGs," in *Proc. of NETGAMES*, Portland, 2004.

[4] H. Lu, "Peer-to-Peer Support for Massively Multiplayer Games," in *Proc. of IEEE INFOCOM*, Hong Kong, 2004.

[5] J. Keller and G. Simon, "Solipsis: A Massively Multi-Participant Virtual World," in *Proc. of PDPTA*, Las Vegas, 2003.

[6] C. GauthierDickey, D. Zappala, *et al.*, "Low Latency and Cheat-Proof Event Ordering for P2P Games," in *Proc. of NOSSDAV*, Ireland, 2004.

[7] A. Rao, K. Lakshminarayanan, *et al.*, "Load Balancing in Structured P2P Systems," in *Proc. of IPTPS*, Berkeley, 2003.

[8] D. Karger and M. Ruhl, "Simple Efficient Load Balancing Algorithms for Peer-to-Peer Systems," in *Proc. of IPTPS*, San Diego, 2004.

[9] J. Byers, J. Considine, *et al.*, "Simple Load Balancing for DHTs," in *Proc. of IPTPS*, Berkeley, 2003.

[10] J. Cernik, "Freewar - MMORPG Browsergame," www.freewar.de.