Peer-to-Peer-based Infrastructure Support for Massively Multiplayer Online Games

Simon Rieche[‡], Marc Fouquet[♯], Heiko Niedermayer[♯],
Leo Petrak[♯], Klaus Wehrle[‡], Georg Carle[♯]

WSI-2006-04
August 2006

[‡]Distributed Systems Group
RWTH Aachen University
{rieche,wehrle}@cs.rwth-aachen.de

[♯]Computer Networks and Internet
University of Tübingen
{fouquet,niedermayer,petrak,carle}@informatik.uni-tuebingen.de

ISSN 0946-3852

Abstract

Online games are an interesting challenge and chance for the future development of the Peer-to-Peer paradigm. Massively multiplayer online games (MMOGs) are becoming increasingly popular today. However, even high-budget titles like World of Warcraft that have gone through extensive beta-testing suffer from downtimes because of hard- and software problems. Our approach is to use structured P2P technology for the server infrastructure of MMOGs to improve their reliability and scalability. Such P2P networks are also able to adapt to the current state of the game and handle uneven distributions of the players in the game world. Another feature of our approach is being able to add supplementary servers at runtime. Our system allows using off-the-shelf PCs as infrastructure peers for participation in different game worlds as needed. Due to the nature of the Economy of Scale the same number of hosts will provide a better service than dedicated servers for each game world.

1 Introduction

Multiplayer games played over the Internet have become very popular in the last few years. An interesting subcategory are the so-called massively multiplayer online games (MMOGs) that allow thousands of player characters to share a single game world. Such a world is usually run on a high-performance and high-availability server cluster. However, even with games that have been extensively beta-tested, downtimes of several hours because of hard- or software failures are not uncommon. Downtimes, especially in the first few weeks after the release, can negatively affect the image of the game and the company that created it.

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

In this paper we use a structured P2P technology for the organization of the infrastructure and thus for the reduction of downtimes in MMOGs. We split the game world in disjunctive rectangular zones and distribute them on different nodes of the P2P network.

Online games are an interesting challenge and chance for the future development of the P2P paradigm. A wide variety of aspects of only theoretically solved and especially yet completely unsolved problems are covered by this application. Security and trust problems appear as well as the need to prevent cheating. The application is not as tolerant to faults as instant messaging or file sharing. Consistent data storage is a problem, decisions and transactions have to be performed in a decentralized way. Moreover, the P2P network is not used as pure lookup service, but more as a communication and application-specific social structure.

The rest of this paper is organized as follows: First we discuss related work in Section 2 and give a brief introduction to P2P and MMOGs and their challenges in Section 3. Section 4 shows our approach to use structured P2P Systems for MMOGs and section 5 the evaluation with player traces from a real MMOG. Finally, Section 6 provides conclusions.

2 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 [11] and [15] the game world is divided into zones, 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, so one always needs backup machines 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, needs to be

saved in a way that makes sure that no information is lost, as players may have invested months or years of work into the game. At the same time one has to prevent players from cheating by modifying this data—which would be possible if it was stored on the player's local harddrives. This suggests that some kind of infrastructure provided by the game manufacturer would still be needed, a full P2P approach does not appear practical. 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 [14] uses a different approach, as it tries to build a P2P network-based on the neighborship 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 [10].

[2] developed a DHT-based system for games with dozens to hundreds of players which is even fast enough for ego-shooters and demonstrated this with a prototype based on the Quake II engine.

Many algorithms exist for load balancing in structured P2P systems [5, 13, 17, 20, 21]. However, most of these systems are not applicable for online games. The virtual server approach [17] is based on the idea of managing multiple partitions of a DHT's address space in one node. Thus, one physical node may act as several independent logical nodes. Each virtual server will be considered by the underlying DHT as an independent node.

3 Background

In this section we give an introduction to the two basic technologies we want to combine: *Massively Multiplayer Online Games* and *structured Peer-to-Peer Systems*.

3.1 Massively Multiplayer Online Games

Massively multiplayer online games (MMOGs) differ from other games mainly in the number of simultaneous players. They allow thousands of players to share a single game world.

This type of games has its origin in the text-based Multi User Dungeons (MUDs) that first appeared in the 1970s. Massively multiplayer games have been very popular in Asia for a number of years, in Europe and the USA they were only played by a minority of players. This changed with the launch of Blizzard's *World of Warcraft* [4] in 2004, which became a worldwide best-seller with more than five million subscribers worldwide.

3.1.1 Massively Multiplayer Online Role Playing Games

Most of today's massively multiplayer titles are role playing games, the term *Massively Multiplayer Online Role Playing Games* (MMORPGs) is well-known in the gaming industry. In this game type, the player's character earns experience while adventuring through the game world. When the player has accumulated a certain amount of experience, she learns new skills or improves old ones.

In early MMORPGs the player mostly earned experience by killing computer-controlled monsters. Today, experience and also other rewards like money or new items are given to the players as a reward for solving quests: A non-player character (NPC) asks the player to perform certain tasks for her. This makes playing a lot more entertaining, as the creators of the game can offer the player a variety of different tasks and also tell a story that evolves with each quest.

Another important aspect of MMORPGs is player-vs-player (PVP) combat on many different scales, from two players that pit their strengths against each other to whole realms or guilds that fight to expand their territories.

In role playing games, combat itself is more a matter of luck and strategy than of quick reflexes. The player has to choose the abilities that she wants to use. Using skills, magic spells or weapons also requires luck, because whether an opponent is hit and how bad he is hit is computed randomly. This is why role playing games are relatively tolerant with respect to delay; it is by far not as important to have a "low ping" as with an ego-shooter.

In a MMORPG context a party is a small group of players that work together for mutual advantage. In many cases quests are designed to be too hard for a single player, so players are forced to join a party and solve the quests together.

The number of players that share a game world at the same time is not only determined by the capabilities of the server cluster, but also by gameplay issues. When a world is crowded by players, there are usually not enough tasks or enemies available for all players to play and fulfill their quests. To give the creators of the game more control over the ratio of enemies to players and therefore over the level of difficulty, some areas are created as instances. This means that each party that enters such an instance plays in its own copy of the area and can not meet other players who play the same region at the same time. Instances break the concept of a massively multiplayer game, in most titles they are used sparingly.

One recently published title named *Guild Wars* [1] is the best example for the use of instances as only small areas of the game world are massively multiplayer while the major part of the game is played in instances. In this game, players can meet in cities, join a party of up to five persons and then leave into the wilderness that surrounds the city. As soon as they leave the city, they enter a part of the world that has been instantiated for them. Therefore one of the players can be the server for the group during this adventure.

3.1.2 Non role-playing games

Other genres than role playing games are by far not as popular in a massively multiplayer context. Real-time strategy games are usually played by 2-8 players, some ego-shooters like Battlefield or Joint Operations may reach 50-100 players. Of course, it is also possible to expand the massively multiplayer paradigm to these genres. This leads to new challenges as ego-shooters are very delay-sensitive and with a massively multiplayer online real-time strategy game the total number of units controlled by the players could get extremely high.

3.1.3 Server Tasks

The central server cluster in a traditional MMOG has to perform various different tasks, including:

- **Management of player positions in the world:** The server has to know the position of each player and distribute this information to other nearby players.
- **Management of the world:** Controlling monsters, non-player characters, treasures, the weather and all other dynamic aspects.
- **Combat:** To avoid cheating, the effects of actions taken by the players in combat should be calculated on a server.
- **Chat:** Usually a global in-game-chat is implemented on the server.
- **Accounting for the players:** The player's characters must be securely saved. This is a task that makes the creation of a real P2P multiplayer game—with all clients acting as servers and without a central infrastructure—very hard.

3.1.4 Challenges

The first and possibly greatest challenge that the server infrastructure of a MMOG faces are the first few days after the game has been launched. An eagerly expected game like *World of Warcraft* can have a huge number of players on the servers after a few days. For such games it is quite common to have different areas for players with different levels of experience. Thus, an additional challenge here is that all those new players will concentrate in the beginner areas of the game and the load on the servers will not be balanced equally.

This is a general problem that one also has to face when the first rush on the game is over. Players do not evenly distribute over the world, they tend to form large groups to raid villages of enemy guilds or realms. Sophisticated mechanisms to balance the load are required here. It would be especially advantageous if one could simply connect another server to the cluster in a high-load condition and the load would be split automatically.

The leading MMORPG in the market—Blizzard's World of Warcraft with over 6 Million players worldwide—is constantly suffering from overloaded servers. Players are complaining about lag, downtimes of the game worlds and long queues when trying to connect to the game. As a consequence, players in China even went on strike on March 15th, 2006 [12].

Another challenge that needs to be addressed is the reliability of the game. Games occasionally experience server instabilities caused by hard- or software failures. In this situation one wishes to have the possibility to simply disconnect a machine that shows a problem, the load should automatically be split up among the remaining servers. Other challenges that are out of our scope in this paper are data management, various hand-overs, ways to guarantee a certain level of consistency, and may more.

3.2 Peer-to-Peer Systems

Peer-to-Peer [16, 22] systems are well-known for their use in filesharing networks. However, Peer-to-Peer technology is the key for many distributed applications. Key features are high scalability, high availability, high redundancy, self-organization, and the reduction of hot spots. However, only structured Peer-to-Peer systems can provide all of these features [3]. In the following sections

we will introduce structured Peer-to-Peer systems, discuss the requirements for massively multi-player games and propose a concept for realization with support for accounting and the conservation of the required security.

3.2.1 Structured Peer-to-Peer Systems

Currently, different strategies for managing and looking up data in Peer-to-Peer systems exist [22]. A rather inefficient solution—mainly used by so-called unstructured Peer-to-Peer approaches—is to store data somewhere and then to flood requests to nearly all systems in the network. In such cases, the P2P system does not scale well, since the network is overloaded with requests.

With structured Peer-to-Peer Systems, like Content Addressable Network (CAN) [18] or Chord [23], a far more efficient method for storing data in a Peer-to-Peer system emerged in the last few years. With their efficient, scalable, and self-organizing algorithms for data retrieval and management, structured Peer-to-Peer Systems offer crucial advantages compared to unstructured approaches like Gnutella.

The basic principle of a structured Peer-to-Peer System is to map data into an address space and distribute the address space in a structured manner among the participating network nodes. Generally, this space is divided into sections which are assigned to individual nodes. Each node is then responsible for managing all data assigned to its section.

3.2.2 Content Addressable Network

The Distributed Hash Table (DHT) based system Content Addressable Network (CAN) [18] uses a d-dimensional identifier spaces for placing data and nodes in a P2P system. Values of data are hashed and mapped onto numerically close nodes in the CAN address space. Since every node only needs knowledge of $2d$ direct neighbors in the identifier spaces locating objects and routing messages is very simple. The identifier space is divided into non-overlapping zones, which are distributed on all nodes. Messages contain the destination coordinates, which can be calculated with a well known hash function, and every node forwards them to the neighbor with the coordinates closest to the destination.

4 Peer-to-Peer-based Infrastructure Support for MMOGs

This section shows our approach to use structured P2P technology for the organization of the infrastructure for MMOGs. The game world is split in disjunctive zones and distributed on different nodes of the P2P network [7, 19].

4.1 Requirements

There are a number of technical requirements our system needs to fulfill. First of all, accounting and charging needs to be possible as computer game companies wish to earn money with their products. In the interest of the players the game has to reliably store character data and prevent cheating. With respect to gameplay, it may not have to support hard real-time conditions. However, it

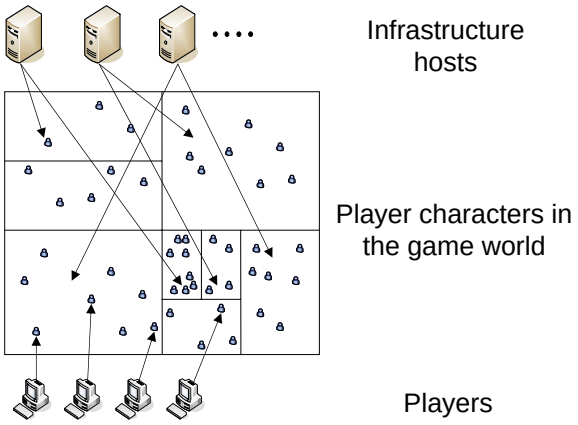


Figure 1: The architecture for the P2P-based infrastructure for MMOGs. Each infrastructure host is responsible for different virtual zones in the world.

has to scale well and keep running smoothly even when load balancing between servers is necessary to enable a good service quality and to react to dynamics of the load, caused e.g. by temporary hot spots in the game world. Since we want to reduce downtimes, it is necessary to support joins and leaves of physical servers. Last but not least, the cost for infrastructure and administration should be low, respectively reduced.

4.2 Architecture

As [15] and [11], we use the fact that only a limited region of the game world is interesting for the player, as his character can only see a limited area and also has a limited movement speed. This allows splitting the game world into regions, whereas the current region and a small number of neighboring regions are in the player's visible range.

The game world is defined by a map, which is split into disjunctive zones in d dimensions (mostly two or three) and distributed on different nodes of the P2P network. As CAN is a structured P2P System based on an d-dimensional coordinate space, it is best-suited for map-based scenarios. Our approach does not use the Distributed Hash Table functionality of CAN, only the functionality of a structured P2P system is needed, like adding nodes or maintenance of the coordinate space. Figure 1 shows a distribution of servers over the map, which is mapped into a zone-based structured P2P system. Since the zones at the edge of the identifier space are connected with the opposite zones for routing and load balancing, also game worlds simulating a terrestrial globe are possible. The players are assigned to the servers according to the position in the game world.

The game world is distributed on a server infrastructure. 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. But our approach could also be used for a super-peer network in a more distributed setting for

more delay-tolerant games.

Besides using a CAN-like address space as a map to store location-specific information, one can also still use it as a traditional DHT to store data that is not directly related to a location in the game world. For example, to realize the global chat functionality that is common with MMOGs, one needs to look up player characters by name and not by their position in the game world. This can be achieved by simply storing a key-value-pair in the CAN Distributed Hash Table.

4.3 Accounting

Accounting can be done via a standard Authentication, Authorization, and Accounting (AAA) [8, 9] server infrastructure. This can be enforced since the game control is not directed to untrusted player peers, but to infrastructure nodes that organize as a Peer-to-Peer network.

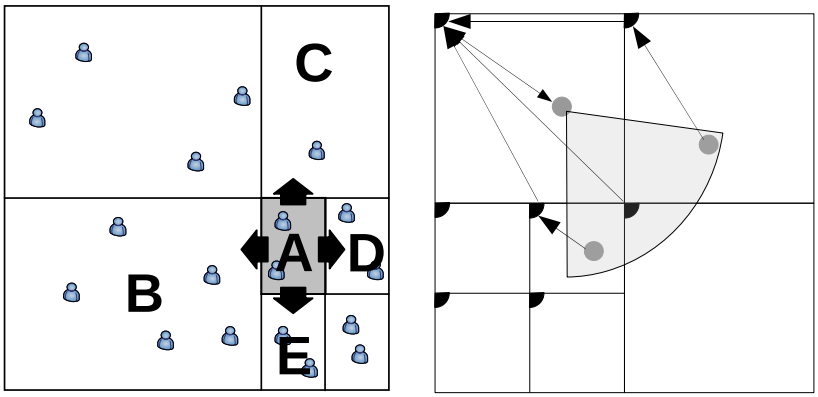
One important aspect of accounting in MMORPGs is saving the players character. As players may accumulate gold, items and experience over years it is very important that this progress is never lost. A player's character can not be saved on her own harddisk, because she may find a way to modify her character to gain an unfair advantage (cheating). Pure Peer-to-Peer based systems without a central infrastructure lack a location where the player's character can be securely saved. As our approach only uses Peer-to-Peer technology on trusted infrastructure nodes, it is no problem to store the character permanently on the game hosts or a dedicated machine and to make backups of this data regularly.

4.4 Availability

Structured P2P systems include mechanisms, that allow neighboring peers to take over the area of a server that has disconnected [18]. However the state information that was stored about this area will be lost, if it has not been replicated somewhere in the network.

As the ratio of disconnects is far lower in our case than with a player P2P network—the game servers should be more reliable than users peers—the replication problem is not as serious as in [15] though. A server that owns area A can simply propagate changes to the game state to one or more other servers $B - E$ that own a neighboring area (cf. Figure 2(a)). It would be appropriate to choose nodes with low load to store the replicas. Note that parts of this information are also useful in $B - E$, as some of players in these areas may be close to the borders and therefore able to see what happens in area A (cf. Figure 2(b)). The other neighbors still receive information that might be interesting for their players, e.g. information about objects and characters that are within visible range of the zone border.

If a failure occurs, the host with the lowest load is chosen among the machines that replicated the information, e.g. the host that owns area D. Since the game state for area A is already known at this node, no further copy is needed to run the game. Then, the load-balancing algorithms presented below can be used to re-distribute the load on the physical servers.



(a) Replication mechanism: Nodes propagate their state to one or more neighbors. If a host suddenly disconnects, a neighbor can take over immediately.

(b) View of a player character: A node has a view across zones borders. The messages are send trough the host, responsible for each zone.

Figure 2: Replication mechanism and view of player characters in the structured P2P infrastructure.

4.5 Pooling

Any P2P approach has to support joins and leaves of nodes. As a consequence, new hosts can be added to the P2P infrastructure when needed. Suitable hosts are not only high-end servers; the game could also run on standard PCs.

On the basis of the virtual server concept hosts may have virtual servers in different game worlds. This has two major implications.

First, the downtime of one or only few servers will probably not be able to stop the game. The availability of the game will be much higher than today. Only when a new version has to be installed on all servers and clients that can not interoperate with the old version there will be a definite stop.

Second, since hosts can be pooled and used for the game or instance that currently needs them most the so-called Economy of Scale[1] comes into play. The consequence is a better service with the same number of resources or a potential to reduce costs.

4.6 Security

The main security functionality can be realized using a Authentication, Authorization, and Accounting (AAA) [8,9] infrastructure. These dedicated AAA servers grant peers access to the world and registered players access to the game.

Since the infrastructure peers are controlled by the game company (or a

[1]The Economy of Scale in this context is also called Multiplexing Gain in the theory of communication networks. It states that an increase in demand does not necessarily need the same increase in capacity to achieve the same service quality.

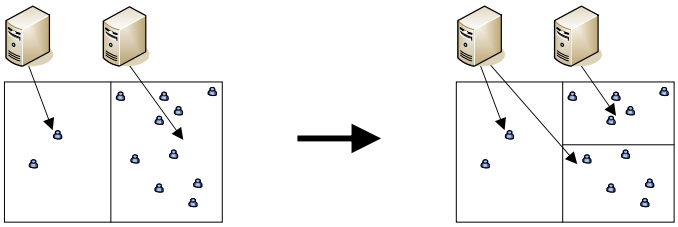


Figure 3: Load Balancing in the P2P-based infrastructure for MMOGs.

community of trusted peers) all nodes in the network are trusted. The peers have to be authenticated since an outside peer could try to enter the network and attack it. Players have to be authenticated, but this, too, can be solved by the AAA infrastructure.

4.7 Load Balancing

Load balancing is an important issue, as one can not expect the players to distribute uniformly in the game world. Some places in the game world, cities for example, can be expected to be more populated than areas in the countryside. This knowledge could be used to statically configure the servers of an MMOG in a way that distributes the expected load evenly. But in many games, it is common to form large groups for raiding enemy territories for example. In such a case, dynamic load balancing is required.

The virtual server approach [17] 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 virtual server will be considered an independent node by the underlying structured P2P System. In our system one virtual server 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 to place and transfer 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 virtual server to another node. Additionally, zones with too many players inside can be split and one part can be sent to another server. Also, adjacent zones with low numbers of players can be merged for a lower number of internal messages.

When an area has to be split because it is overpopulated, one of five algorithms is used to decide how the split should be performed (cf. Fig. 3).

- **SplitCenter** splits areas along the center horizontally or vertically in exchange. If area A was split into areas B and B' horizontally, then area B will be split into C and C' vertically.
- **MaxDistToBorders** also splits along the center. This algorithm decides whether to split horizontally or vertically by maximizing the average dis-

tance of the players to the newly created border. The idea behind this is, to minimize internal traffic as looking and walking over borders always creates overhead.

- **IntelliDistance** again splits along the center and decides whether to split horizontally or vertically, such that as few players as possible can see the other side of the new border. The rationale behind this is the same as with MaxDistToBorders.
- **EqualNumbers** splits along the center and decides whether to split horizontally or vertically in a way that makes the number of players in the new regions is as equal as possible. The expectation is that this algorithm will perform better than SplitCenter in terms of load-balancing.
- **VarAreas** splits horizontally or vertically as SplitCenter does. However it places the border not along the center but at the centroid of the players. As with EqualNumbers the expected result is better load-balancing.

5 Evaluation

In this section we evaluate our approach with respect to the load balancing issues. First, we briefly introduce the simulation methodology. Secondly, we discuss how players move in the game world and finally present our simulation results.

The two major criteria for the evaluation are:

- **The variance of the physical server load** is the variance of "computational" load we determined for the various physical servers. Ideal would be 0, i.e. same load for all.
- **The number of internal messages** is a way to measure the overhead introduced by distributing the game world on multiple servers. Internal messages are exchanged between physical servers, they are caused by players looking or walking over area borders and when players are assigned to another server because a virtual server has been moved.

5.1 Player Behavior

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

- **Random walk trace data** is a collection of generic data represents movement of users according to the Random Walk Mobility Model. It covers up to 300 users with four hours of movement time.
- **Random waypoint trace data** has the same properties by using the Random Waypoint Mobility Model.
- **Freewar trace data** consists of real player-movements traced in Freewar [6] over a period of up to five hours (cf. Fig. 4). Approximately 400 players were online in this period of time, the number of concurrent players varies over time since users join or leave the game. So there is a dynamic number

Figure 4: Map of the MMOG Freewar.

of players over time. Also the Freewar traces show an extremely uneven distribution of the players over the world with hot spots in cities.

5.2 Results

Figure 5 shows the coefficient of variation (CV) of the physical server load, with all five algorithms and with real player-traces from Freewar. One can see that *IntelliDistance* performs best, *SplitCenter* is the second-best algorithm. *EqualNumbers* and *VarAreas* have a very high CV. The reason for this is that those algorithms already achieve a relatively good load-balancing with few split operations, so the game-world is not split into as many areas as with the other algorithms. Figure 5 shows also the CV for artificial "Random Walk" and "Random Waypoint" traces. One can see that the CV is generally much lower here, as the players are distributed more evenly. Again *EqualNumbers* and *VarAreas* have the highest CV, this time *SplitCenter* was the best algorithm.

In Figure 6 one can see the number of internal messages caused by distributing the game world on multiple peers. Internal messages are triggered by players that are close to an area-border so they can see the other side, by players crossing an area border (handover) and by virtual servers that are moved from one physical server to the next.

The algorithms *MaxDistToBorder* and *IntelliDistance* perform approximately as good as *SplitCenter* with the real player-traces from Freewar. Splitting areas in a way that minimizes the number of players who can see beyond the border in the moment of the split is ineffective. The algorithms *EqualNumbers* and *VarAreas* perform especially well here. At the first glance it appears strange

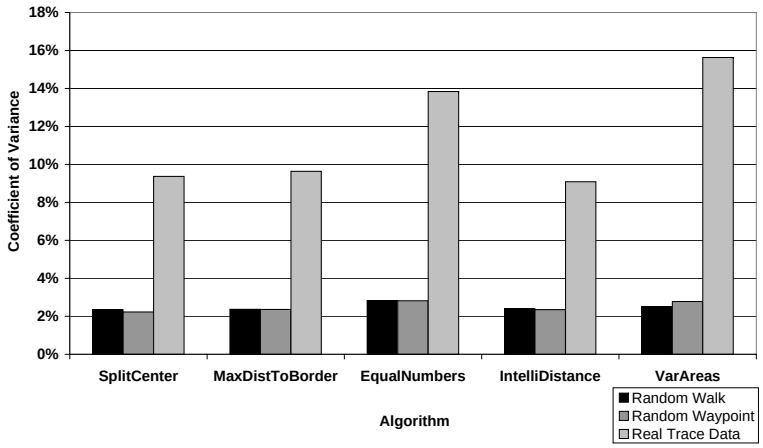


Figure 5: The effectiveness of the load-balancing algorithms with real trace data of the MMOG Freewar and artificial trace data.

that the algorithms designed for good load-balancing are good with regard to the number of internal messages while they appeared bad regarding the coefficient of variation above. The explanation is that these algorithms split the world more effectively, so fewer splits are required to prevent servers from being overloaded. Therefore the game world is split into fewer areas which reduces the number of internal messages.

Figure 6 also shows the number of internal messages with artificial "Random Walk" and "Random Waypoint" traces. Surprisingly *EqualNumbers* and *VarAreas*, the algorithms with the best results with real player data, perform specially bad here. In the artificial traces, the players are distributed on the map evenly, therefore those two algorithms gain little advantage from their power to split areas. However they have disadvantages when merging regions as - specially with variable areas - the chances of having a common border with the neighboring area are reduced.

This shows also that artificial trace data is of little use for evaluation of MMOGs. Real players concentrate at certain hot spots on the map, they move in groups and sometimes even show a "flash crowd"-like behavior. Therefore "Random Walk" and "Random Waypoint" do not generate results that would be applicable to real games.

In a real MMOG, one may want to start splitting the world using the *SplitCenter* algorithm, as it produces good results and appears to be robust against changes of the player distribution. *VarAreas* is even more effective and may be the algorithm of choice if the player distribution is rather uneven and shows hot-spots. But switching between algorithms is also possible at run-time.

6 Conclusions and Future Work

In this paper we use a structured P2P technology for the organization of the infrastructure and thus for the reduction of downtimes in MMOGs. In our approach we split the game world in disjunctive zones similar to the CAN Dis-

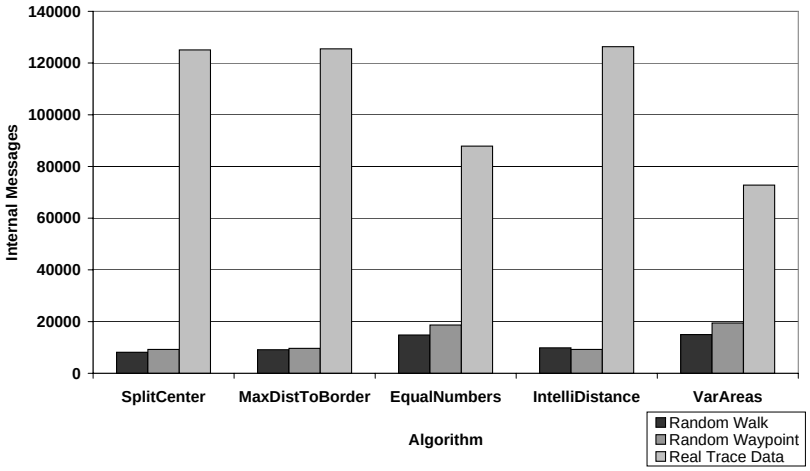


Figure 6: The number of internal messages with real trace data of the MMOG Freewar and artificial trace data.

tributed Hash Table and distribute them 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 accordingly 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.

We will continue working on P2P support for MMOGs. Future work includes more a detailed investigation of handovers and state consistency between the servers and even better load-balancing algorithms.

References

[1] ArenaNet Inc. Guild Wars. http://www.guildwars.com/.

[2] Jeffrey Pang Ashwin Bharambe and Srinivasan Seshan. A Distributed Architecture for Multiplayer Games. In *Proc. of ACM/USENIX NSDI 2006*, San Jose, USA, 2006.

[3] Hari Balakrishnan, M. Frans Kaashoek, David Karger, Robert Morris, and Ion Stoica. Looking Up Data in P2P Systems. *Communications of the ACM*, 46(2):43–48, 2003.

[4] Blizzard Entertainment. World of Warcraft. www.worldofwarcraft.com.

[5] John W. Byers, Jeffrey Considine, and Michael Mitzenmacher. Simple Load Balancing for Distributed Hash Tables. In *Proc. of 2nd International Workshop on Peer-to-Peer Systems (IPTPS)*, Berkeley, CA, 2003.

[6] Jirko Cernik. Freewar - MMORPG Browsergame. www.freewar.de.

[7] Jirko Cernik. Peer-to-Peer-Infrastruktur für Massive Multiplayer Online Role-Playing Games (MMORPGs). Diploma thesis, University of Tübingen, Germany, 2006.

[8] Cees de Laat, George Gross, et al. Generic AAA Architecture. IETF, RFC 2903, 2000.

[9] S. Farrell, J. Vollbrecht, et al. AAA Authorization Requirements. IETF, RFC 2906, 2000.

[10] Chris GauthierDickey, Daniel Zappala, Virginia Mary Lo, and James Marr. Low Latency and Cheat-Proof Event Ordering for P2P Games. In *Proc. of 14th International Workshop on Network and Operating System Support for Digital Audio and Video (NOSSDAV)*, Ireland, 2004.

[11] Takuji Iimura, Hiroaki Hazeyama, and Youki Kadobayashi. Zoned federation of game servers: a Peer-to-Peer approach to scalable multi-player online games. In *Proceedings of the 3rd Workshop on Network and System Support for Games (NETGAMES)*, Portland, OR, 2004.

[12] Interfax China. Gamers threaten mass protest against The9's operation of WoW in China, 2006.

[13] David Karger and Mathias Ruhl. Simple Efficient Load Balancing Algorithms for Peer-to-Peer Systems. In *Proc. of 3rd International Workshop on Peer-to-Peer Systems (IPTPS)*, San Diego, CA, 2004.

[14] Joaquin Keller and Gwendal Simon. Solipsis: A Massively Multi-Participant Virtual World. In *Proceedings of the International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA)*, Las Vegas, NV, 2003.

[15] Honghui Lu. Peer-to-Peer Support for Massively Multiplayer Games. In *Proc. of the 23rd Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM)*, Hong Kong, China, 2004.

[16] Dejan S. Milojicic, Vana Kalogeraki, Rajan Lukose, Kiran Nagaraja, Jim Pruyne, Bruno Richard, Sami Rollins, and Zhichen Xu. Peer-to-Peer Computing. Technical Report HPL-2002-57, HP Laboratories, Palo Alto, 2002.

[17] Ananth Rao, Karthik Lakshminarayanan, Sonesh Surana, Richard M. Karp, and Ion Stoica. Load Balancing in Structured P2P Systems. In *Proc. of 2nd International Workshop on Peer-to-Peer Systems (IPTPS)*, Berkeley, CA, 2003.

[18] Sylvia Ratnasamy, Paul Francis, Mark Handley, Richard M. Karp, and Scott Shenker. A Scalable Content-Addressable Network. In *Proc. of the ACM SIGCOMM 2001 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication*, San Diego, CA, 2001.

[19] Simon Rieche, Jirko Cernik, Marc Fouquet, Heiko Niedermayer, Klaus Wehrle, and Georg Carle. Peer-to-Peer-based Infrastructure Support for Massively Multiplayer Online Games. In *Poster at Dagstuhl Seminar 06131 "Peer-to-Peer Systems and Applications"*, Dagstuhl, Germany, 2006.

[20] Simon Rieche, Leo Petrak, and Klaus Wehrle. A Thermal-Dissipation-based Approach for Balancing Data Load in Distributed Hash Tables. In *Proc. of 29th Annual IEEE Conference on Local Computer Networks (LCN)*, Tampa, FL, 2004.

[21] Simon Rieche, Leo Petrak, and Klaus Wehrle. Comparison of Load Balancing Algorithms for Structured Peer-to-Peer Systems. In *Proc. of Workshop on Algorithms and Protocols for Efficient Peer-to-Peer Applications 2004*, Ulm, Germany, 2004.

[22] Ralf Steinmetz and Klaus Wehrle, editors. *Peer-to-Peer Systems and Applications*, volume 3485 of *Lecture Notes in Computer Science, LNCS*. Springer, Heidelberg, Germany, 2005.

[23] Ion Stoica, Robert Morris, David Karger, M. Frans Kaashoek, and Hari Balakrishnan. Chord: A Scalable Peer-to-peer Lookup Service for Internet Applications. In *Proc. of the ACM SIGCOMM*, San Diego, CA, 2001.